

4D-A131 715

JOINT LOGISTICS-OVER-THE-SHORE (LOTS) MAIN TEST
AUTOMATED DATA BASE REDUCTION PROGRAMS(U) ORI INC
SILVER SPRING MD H CASEY ET AL. MAR 79 ORI-TR-1477

1/6

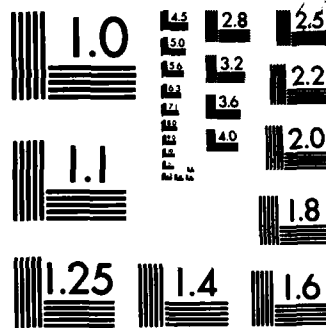
UNCLASSIFIED

MDA903-75-C-0016

F/G 9/2

NL

ORI

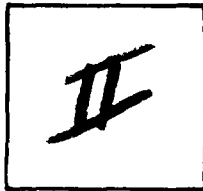


MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

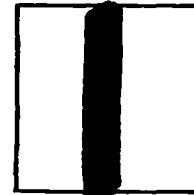
PHOTOGRAPH THIS SHEET

ADA131715

DTIC ACCESSION NUMBER



LEVEL



INVENTORY

Rept. No. ORI-TR-1477

DOCUMENT IDENTIFICATION Final, Nov.'77-Nov.'78

Contract MDA-903-75-C-0016

Mar. 79

DISTRIBUTION STATEMENT A

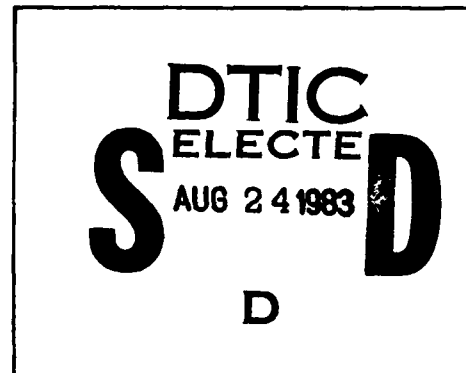
Approved for public release;
Distribution Unlimited

416

DISTRIBUTION STATEMENT

ACCESSION FOR	
NTIS	GRA&I <input checked="" type="checkbox"/>
DTIC	TAB <input type="checkbox"/>
UNANNOUNCED	<input type="checkbox"/>
JUSTIFICATION	
BY	
DISTRIBUTION /	
AVAILABILITY CODES	
DIST	AVAIL AND/OR SPECIAL
A	

DISTRIBUTION STAMP



DATE ACCESSIONED

83 08 2 . 078

DATE RECEIVED IN DTIC

PHOTOGRAPH THIS SHEET AND RETURN TO DTIC-DDA-2

ADA131715

ORI

DISTRIBUTION STATEMENT A

Approved for public release
Distribution Unlimited

ORI 1400 SPRING ST.
Silver Spring, Maryland 20910

JOINT LOGISTICS-OVER-THE-SHORE (LOTS)
MAIN TEST AUTOMATED DATA
BASE REDUCTION PROGRAMS

MARCH 1979

PREPARED FOR:
OFFICE OF THE SECRETARY OF DEFENSE
OFFICE OF THE UNDER SECRETARY OF DEFENSE,
RESEARCH AND ENGINEERING
OFFICE OF THE DEPUTY DIRECTOR, TEST AND EVALUATION
WASHINGTON, DC 20301

DISTRIBUTION STATEMENT A

Approved for public release
Distribution Unlimited

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ORI, Inc. TR 1477	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) JOINT LOGISTICS-OVER-THE-SHORE (LOTS) MAIN TEST AUTOMATED DATA BASE REDUCTION PROGRAMS		5. TYPE OF REPORT & PERIOD COVERED Documentation - Final Report November 1977 - November 1978
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) H. Casey, A. Green		8. CONTRACT OR GRANT NUMBER(s) MDA-903-75-C-0016
9. PERFORMING ORGANIZATION NAME AND ADDRESS ORI, Inc. 1400 Spring Street Silver Spring, MD 20910		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of The Secretary of Defense OUSDR&E, ODDT&E, Tactical Air & Land Warfare Sys. The Pentagon, Washington, DC 20301		12. REPORT DATE March 1979
		13. NUMBER OF PAGES 334 pages
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) N/A		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) Contracting Officer's Technical Representative - Mr. Richard Ledesma		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		
Amphibious	LOTS	LARC-LX Merchant
Barge-TCDF	DeLong	LCM8 Ships
Cargo	Elevated Causeway	Logistics Throughput
Containerization	LACH	Logistics-Over-The-Shore Transportation
Crane-on-deck	LACV-30	LOTS
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)		
<p>This report describes the automated data base and the programs used for consistency checks and data reduction with Joint Logistics-Over-the-Shore (LOTS) Test and Evaluation Program. The test was conducted at Fort Story, Va. in August, 1977, over a 15-day period. The data were collected by the Joint LOTS Test Directorate.</p> <p>Timings were conducted of containership discharge operations using the barge-temporary container discharge facility and crane-on-deck methods.</p>		

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 68 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. (CONT)

Containers (individually identified) were lightered to and from ship and shore by LCUs, LCM8s, LARC-LXs, LARC-XVs, LACV-30s, and causeway ferries. Shoreside container transfer facilities included a 300-ton crane on a jetty, a 140-ton amphibian discharge crane, an elevated causeway system, lightweight amphibious container handler, and a DeLong pier system. Transfer operations between lighters, cranes, materials handling equipment, and tractor-trailers were identified by individual craft and timed for both container off-load and retrograde operations. The final destination for containers was an inland marshaling yard where arrival/departure and off-load/load events were recorded.

Operations were divided into four distinct phases with each phase consisting of off-load and retrograde periods, and having varying units and equipment. Phase I was an Army bare beach, quick reaction, non-mobilization scenario. Phase II simulated the activities involving a Navy-Marine Corps off-load of an assault follow-on echelon (AFOE). Phase III was a joint operation under mobilization conditions and involved improved beach facilities. Phase IV (also referred to as Phase I(R)) involved a one-day repeat of Phase I in order to assess the effects of more experienced crews.

The report mentions voids in the data base and the effects of timing and accuracy of results when analyzing brief events of only 1-2 min. or less. The report cautions that future users of the Joint LOTS Data Base should incorporate error checking routines in any analyses programs in order to correct and/or discard data inconsistencies.

The programs described in this report were used in the analyses of the Joint LOTS test final report (ORI, Inc. TR 1412). The programs and the data base were copied to magnetic tape and sent to the Joint Test Director at Ft. Eustis for retention by the Commandant, US Army Transportation School.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

TABLE OF CONTENTS

		<u>Page</u>
I.	INTRODUCTION	1-1
	BACKGROUND	1-1
	REPORT ORGANIZATION.	1-1
	PROGRAM USES	1-1
	ACCESS	1-2
	LOTS DATA BASE	1-2
II.	CONSISTENCY CHECK PROGRAM.	2-1
	INTRODUCTION	2-1
	PROGRAM DESCRIPTION.	2-1
	SYSTEM 2000 RETRIEVALS FOR THE MATCHR1 FILES	2-5
	MAIN PROGRAM MATCHR1	2-8
	MAIN MATCHR1 FUNCTION FDISCID (1).	2-9
	MAIN MATCHR1 FUNCTION SCATTAB (1).	2-10
	MAIN MATCHR1 SUBROUTINE GDRDATA (2).	2-11
	MAIN MATCHR1 FUNCTION READTBL (2.1).	2-12
	MAIN MATCHR1 FUNCTION NSSCCHK (3).	2-13
	MAIN MATCHR1 SUBROUTINE TBLDUMP (4).	2-14
	MAIN MATCHR1 SUBROUTINE DMPSTER (4.1).	2-15
	MAIN MATCHR1 FUNCTION VLDCGID (5).	2-16

TABLE OF CONTENTS (CONT'D)

		<u>Page</u>
	COMMON BLOCK CDVLC	2-17
	COMMON BLOCK COVLT	2-17
	COMMON BLOCK CGTUNIT	2-17
	COMMON BLOCK CRVLC	2-17
	COMMON BLOCK CRVLT	2-17
	COMMON BLOCK DRTERRS	2-17
	COMMON BLOCK NORMAN1	2-18
	COMMON BLOCK NREPORT	2-18
	COMMON BLOCK NSSCDUP	2-18
	COMMON BLOCK PAGENUM	2-19
	COMMON BLOCK TBLUNIT	2-19
	COMMON BLOCK TITLE	2-19
III.	LIGHTER CYCLE TIMES PROGRAM.	3-1
	INTRODUCTION	3-1
	PROGRAM DESCRIPTION.	3-2
	SYSTEM 2000 RETRIEVALS FOR THE SHIP FILES.	3-5
	SYSTEM 2000 RETRIEVALS FOR THE SHORE FILES	3-6
	MAIN PROGRAM SHPGEN.	3-9
	MAIN SHPGEN SUBROUTINE READ (1).	3-11
	MAIN SHPGEN SUBROUTINE COMPARE (2)	3-13
	MAIN SHPGEN SUBROUTINE SELECT (3).	3-14
	MAIN SHPGEN SUBROUTINE CONVERT (4)	3-15
	MAIN SHPGEN SUBROUTINE RPTINC (5).	3-16
	MAIN SHPGEN SUBROUTINE RPTCOM (6).	3-18
	MAIN SHPGEN SUBROUTINE STOP (7).	3-19
	MAIN PROGRAM SHRGEN.	3-23
	MAIN SHRGEN SUBROUTINE (1)	3-25
	MAIN SHRGEN SUBROUTINE COMPARE (2)	3-27

TABLE OF CONTENTS (CONT'D)

	<u>Page</u>
MAIN SHRGEN SUBROUTINE SELECT (3)	3-28
MAIN SHRGEN SUBROUTINE COMPUTE (4).	3-29
MAIN SHRGEN SUBROUTINE CONVERT (4.1).	3-31
MAIN SHRGEN SUBROUTINE RPTINC (5)	3-32
MAIN SHRGEN SUBROUTINE RPTCOM (6)	3-34
MAIN SHRGEN SUBROUTINE STOP (7)	3-36
MAIN PROGRAM PREV	3-39
MAIN PREV SUBROUTINE CONVERT (1).	3-40
MAIN PROGRAM COMBINE.	3-45
MAIN COMBINE SUBROUTINE READ (1).	3-47
MAIN COMBINE SUBROUTINE COMPARE (2)	3-49
MAIN COMBINE SUBROUTINE SELECT (3).	3-50
MAIN COMBINE SUBROUTINE COMPUTE (4)	3-51
MAIN COMBINE SUBROUTINE CONVERT (4.1)	3-53
MAIN COMBINE SUBROUTINE UPDATE (5).	3-54
MAIN COMBINE SUBROUTINE RPTINC (6).	3-56
MAIN COMBINE SUBROUTINE RPTCOM (7).	3-57
MAIN COMBINE SUBROUTINE STOP (8).	3-59
MAIN PROGRAM WRTRPT	3-62
MAIN WRTRPT SUBROUTINE ACCID (1).	3-63
MAIN WRTRPT SUBROUTINE ACCTY (2).	3-65
MAIN WRTRPT SUBROUTINE INID (3)	3-67
MAIN WRTRPT SUBROUTINE INTY (4)	3-68
MAIN WRTRPT SUBROUTINE PBRK (5)	3-70
MAIN WRTRPT SUBROUTINE PRNIND (6)	3-73
MAIN WRTRPT SUBROUTINE PRNIST (7)	3-74
MAIN WRTRPT SUBROUTINE PRNTST (8)	3-75

TABLE OF CONTENTS (CONT'D)

	<u>Page</u>
IV. LIGHTER AND CARGO CYCLE TIMES PROGRAM	4-1
INTRODUCTION.	4-1
PROGRAM DESCRIPTION	4-2
SYSTEM 2000 RETRIEVAL FOR THE LIGHTER/CARGO FILES	4-5
SYSTEM 2000 RETRIEVALS FOR THE CARGO TIMES FILE	4-6
MAIN PROGRAM SHPGEN	4-9
MAIN SHPGEN SUBROUTINE READ (1)	4-11
MAIN SHPGEN SUBROUTINE COMPARE (2).	4-13
MAIN SHPGEN SUBROUTINE SELECT (3)	4-14
MAIN SHPGEN SUBROUTINE CONVERT (4).	4-15
MAIN SHPGEN SUBROUTINE RPTINC (5)	4-16
MAIN SHPGEN SUBROUTINE RPTCOM (6)	4-18
MAIN SHPGEN SUBROUTINE STOP (7)	4-19
MAIN PROGRAM PREV	4-22
MAIN PREV SUBROUTINE CONVERT (1).	4-23
MAIN PROGRAM STEP 8	4-26
MAIN STEP 8 SUBROUTINE READ (1)	4-28
MAIN STEP 8 SUBROUTINE COMPARE (2).	4-29
MAIN STEP 8 SUBROUTINE SELECT (3)	4-31
MAIN STEP 8 SUBROUTINE RPTINC (4)	4-32
MAIN STEP 8 SUBROUTINE RPTCOM (5)	4-33
MAIN STEP 8 SUBROUTINE STOP (6)	4-34
MAIN PROGRAM CARGOID.	4-37
MAIN CARGOID SUBROUTINE READ (1).	4-39
MAIN CARGOID SUBROUTINE CHANGE (2).	4-40
MAIN CARGOID SUBROUTINE BUILD (3)	4-41
MAIN CARGOID SUBROUTINE CONVERT (4)	4-43
MAIN CARGOID SUBROUTINE RPTINC (5).	4-44
MAIN CARGOID SUBROUTINE RPTCOM (6).	4-45

TABLE OF CONTENTS (CONT'D)

	<u>Page</u>
MAIN PROGRAM STEP 13	4-48
MAIN STEP 13 SUBROUTINE COMPUTE (1).	4-50
MAIN STEP 13 SUBROUTINE SAVE	4-51
MAIN PROGRAM STEP 15	4-54
MAIN STEP 15 SUBROUTINE READ (1)	4-56
MAIN STEP 15 SUBROUTINE COMPARE (2).	4-58
MAIN STEP 15 SUBROUTINE SETUP (3).	4-59
MAIN STEP 15 SUBROUTINE SELECT (4)	4-61
MAIN STEP 15 SUBROUTINE CONVERT (5).	4-62
MAIN STEP 15 SUBROUTINE RPTINC (6)	4-63
MAIN STEP 15 SUBROUTINE RPTCOM (7)	4-65
MAIN STEP 15 SUBROUTINE STOP (8)	4-67
MAIN PROGRAM WRTRPT.	4-70
MAIN WRTRPT SUBROUTINE ACCTY (1)	4-72
MAIN WRTRPT SUBROUTINE INTY (2).	4-74
MAIN WRTRPT SUBROUTINE PBBRK (3)	4-75
MAIN WRTRPT SUBROUTINE PRNIND (4).	4-77
MAIN WRTRPT SUBROUTINE PRNTST (5).	4-79
MAIN WRTRPT SUBROUTINE CHECK (6)	4-80
V. CRANE CYCLE TIMES PROGRAM.	5-1
INTRODUCTION	5-1
PROGRAM DESCRIPTION.	5-1
SYSTEM 2000 RETRIEVALS FOR PROGRAM MAGIC	5-3
MAIN PROGRAM MAGIC	5-6
MAIN MAGIC SUBROUTINE CONVERT (1).	5-7
MAIN PROGRAM CYCLOPS	5-10
PROGRAM CRDBINT (1).	5-12
PROGRAM CRDBINT FUNCTION RDCARDF (1.1)	5-13
PROGRAM CRDBINT FUNCTION WGHTCAT (1.11).	5-14

TABLE OF CONTENTS (CONT'D)

	<u>Page</u>
PROGRAM CRDBINT FUNCTION WCATGET (1.111)	5-15
PROGRAM CRDBINT FUNCTION CVOPRID (1.12).	5-16
PROGRAM CRDBINT FUNCTION CLASSIFY (1.13)	5-17
PROGRAM CRDBINT FUNCTION FCLTYPE (1.131)	5-18
PROGRAM CRDBINT FUNCTION XPTTYPE (1.132)	5-19
PROGRAM CRSCINT (2).	5-20
PROGRAM CRSCINT FUNCTION MINUTES (2.1)	5-21
PROGRAM CRSCINT FUNCTION BYCODE (2.2).	5-22
PROGRAM CRSCINT FUNCTION BGTBLOK (2.3)	5-23
PROGRAM CRSCINT FUNCTION WGTMSKR (2.4)	5-24
PROGRAM RPTORTR (3).	5-25
PROGRAM RPTORTR FUNCTION TBLMAKR (3.1)	5-26
PROGRAM RPTORTR SUBROUTINE TBLDMPR (3.2)	5-27
PROGRAM RPTORTR FUNCTION GETSLCR (3.3)	5-28
PROGRAM REPORTR FUNCTION SELECT (3.4).	5-29
PROGRAM REPORTR SUBROUTINE MKSCWRD (3.5)	5-30
PROGRAM REPORTR FUNCTION WCIDENT (3.51).	5-31
PROGRAM REPORTR SUBROUTINE CVRPCCD (3.6)	5-32
PROGRAM REPORTR SUBROUTINE WRITRPT (3.7)	5-33
PROGRAM REPORTR SUBROUTINE STATS (3.71).	5-34
COMMON BLOCK BLANK	5-35
COMMON BLOCK BST	5-35
COMMON BLOCK NSCREC.	5-35
COMMON BLOCK SHFSTAT	5-35
COMMON BLOCK WGTCOM.	5-35
APPENDIX A - CONSISTENCY CHECK PROGRAM LISTINGS	A-1
APPENDIX B - LIGHTER CYCLE TIMES PROGRAM LISTINGS	B-1
APPENDIX C - LIGHTER AND CARGO CYCLE TIMES PROGRAM LISTINGS	C-1
APPENDIX D - CRANE CYCLE TIMES PROGRAM LISTINGS	D-1

LIST OF FIGURES

1.1	LOTS Data Base Structure	1-3
2.1	Consistency Check Program Sample Output For the Elevated Causeway Crane Activities for Shift 2, 11 August 1978. (From the C1300 Repeating Group of the LOTS Data Base)	2-2
2.2	MATCHR1 Sample Output for the Elevated Causeway Discharging Lighter Activities for Shift 2, 11 August 1978. (From the C1500 Repeating Group of the LOTS Data Base)	2-3
2.3	MATCHR1 Sample Output for the Elevated Causeway Receiving Land Carrier Activities for Shift 2, 11 August 1978. (From The C2000 Repeating Group of the LOTS Data Base)	2-4
2.4	MATCHR1 Structure	2-6
2.5	Main Program (MATCHR1) Flow Diagram	2-7
3.1	Lighter Cycle Times Program Sample Output	3-3
3.2	Flow Chart of the Steps in the Lighter Cycle Times Program	3-4
3.3	SHPGEN Structure	3-7
3.4	Main Program (SHPGEN) Flow Diagram	3-8
3.5	SHRGEN Structure	3-20
3.6	Main Program (SHRGEN) Flow Diagram	3-21
3.7	(Cont'd) Main Program (SHRGEN) Flow Diagram	3-22
3.8	PREV Structure	3-37
3.9	Main Program (PREV) Flow Diagram	3-38
3.10	Combine Structure	3-41
3.11	Main Program (SHRGEN) Flow Diagram	3-42
3.12	(Con't) Main Program (SHRGEN) Flow Diagram	3-43
3.13	(Con't) Main Program (SHRGEN) Flow Diagram	3-44
3.14	WRTRPT Structure	3-60
3.15	Main Program (WRTRPT) Flow Diagram	3-61
4.1	Lighter and Cargo Cycle Times Program Sample Output	4-3
4.2	Flow Chart of the Steps in the Lighter and Cargo Cycle Times Program	4-4
4.3	SHPGEN Structure	4-7
4.4	Main Program (SHPGEN) Flow Diagram	4-8
4.5	PREV Structure	4-20
4.6	Main Program (PREV) Flow Diagram	4-21

LIST OF FIGURES (CON'T)

Figure

4.7	STEP8 Structure	4-24
4.8	Main Program (STEP8) Flow Diagram	4-25
4.9	Cargo ID Structure	4-35
4.10	Main Program (Cargo ID) Flow Diagram	4-36
4.11	Step 13 Structure	4-46
4.12	Main Program (Step 13) Flow Diagram	4-47
4.13	Step 15 Structure	4-52
4.14	Main Program (Step 15) Flow Diagram	4-53
4.15	WRTTRPT Structure	4-68
4.16	Main Program (WRTTRPT) Flow Diagram	4-69
5.1	Crane Cycle Times Program Sample	5-2
5.2	Magic Structure	5-4
5.3	Main Program (Magic) Flow Diagram	5-5
5.4	CYCLOPS Structure	5-8
5.5	Main Program(CYCLOPS) Flow Diagram	5-8

I. INTRODUCTION

BACKGROUND

Described in this report are the Logistics-Over-the-Shore (LOTS) data reduction programs that were developed for the Director, Defense Test and Evaluation, ODDR&E, OSD under modification number 9 to Contract No. MDA-903-75-C-0016.

REPORT ORGANIZATION

Each data reduction program is described in a separate chapter of this report. Each chapter begins with an introduction which states what the program computes and gives a sample output. Then an overview is given on how the program is designed. Included in this section for each multi-step program is a flow chart of the related steps. Following this is a description of the System 2000 data management system retrieval programs. Finally, structure diagram, flow chart and description of each main program, and sub-routine and function for data base reduction is presented.

Contained in each Appendix is a listing of the System 2000 retrieval statements, control statements needed to execute the program, and a listing of the program.

PROGRAM USES

Consistency Check Program

The outputs from this program flag invalid container identification numbers by comparison with a control list; indicate a "no match" whenever there is an inconsistency in the data relating to a container transfer from ship to lighter, lighter to beach facility and beach facility to truck, and

finally from truck to a storage location in the marshaling yard. The consistency check program does not check every data entry, but rather only those described above. The results of this program were used to make corrections to the data base whenever the data could be verified by reference to data collection sheets. Corrections were possible in most cases. After the corrections were applied, the consistency check program was rerun and a substantial reduction in invalid cargo identifiers and "no matches" was noted.

Lighter Cycle Times Program

This program traces each lighter carrying containers between ship and shore facilities. It computes the time the lighter spent at the ship, time the lighter was underway, the time the lighter spent at the shore facility and lighter succession times. It also gives average and standard deviation values for these computed times. This program is useful in analyzing segments of lighter cycles.

Lighter and Cargo Cycle Times Program

The Lighter and Cargo Times program provides a basis for analyzing interaction of lighter and ship crane container transfer operations. It computes the component parts of the lighter and cargo cycles at that operating interface. The results of this program are useful in analyzing lighter activities at the ship as a part of the total lighter cycle analysis.

Crane Cycle Times Program

For selected crane cycles, this program computes the mean, standard deviation, median and, as an option, a histogram. Selection criteria such as direction of container movement, container weight, crane operator ID are inputs to the program. The Crane Cycle Times Program is useful in analyzing crane cycles and the effects of selected parameters on cycle times in sensitivity analysis, learning curves, and the like.

ACCESS

The programs and data base described in this report have been copied to magnetic tape and sent to the Joint Test Director at Ft. Eustis, Virginia for file. Anyone wishing to use these programs should submit a request to the Commandant, The US Army Transportation School, Ft. Eustis, Virginia.

LOTS DATA BASE

The LOTS Data Base is described in a separate report.¹ For ready reference, the data base structure is given in Figure 1.1.

¹ The BDM Corporation, Logistics-Over-the-Shore (LOTS) Program Documentation and User Information Report, BDM/N-78-057-TR, 10 January 1978.

It should be noted that there are significant voids in the Joint LOTS data base. Data collection sheets have been reviewed in detail to reduce data voids to a minimum. The remaining areas with significant voids occurred as a result of shift changes and periods of prolonged inactivity, as for example, in the marshaling yard at night when data collection efforts would stop and intermittent container arrivals from the beach would be missed. An additional problem relates to unsynchronized watches used for timing events at each site. The differences vary from a minute to as high as eight or ten minutes. Thus, care must be taken in analyzing transit times of container movements from ship to shore, beach to marshaling yard and return. Likewise, the times for containers to clear a facility (e.g., from lighter to beach, beach to truck, truck to depart) must be constructed from separate segments of time to avoid watch synchronization errors. (As many as four data collectors timed the lighter, crane, frontloader and truck activities at a beach site.) Finally, some errors in event times remain randomly scattered in the data base which were incorrectly recorded by the data collector or inadvertently introduced during the coding or key punching process. These errors become apparent when using programs for analyzing crane cycles, lighter cycles, etc. The subtraction of times to compute time segments for successive events pinpoint such errors by giving negative results or unrealistically long periods of time. The most obvious of these are multiples of hours (or 60 min.).

Future users of the Joint LOTS Data Base should incorporate error checking routines in any analyses programs developed to correct and/or discard data inconsistencies.

II. CONSISTENCY CHECK PROGRAM

INTRODUCTION

The function of this program is to perform a consistency check among selected components within the cargo activities, lighter activities and land carrier activities repeating group structures of the LOTS SYSTEM 2000 data base.

The outputs flag invalid container identification numbers by comparison with a control list; indicate a "no match" whenever there is an inconsistency in data relating to a container transfer from ship to lighter, lighter to beach facility, and beach facility to truck, and finally from truck to a storage location in the marshaling yard.

A sample output is given in Figures 2.1, 2.2, and 2.3.

PROGRAM DESCRIPTION

The Consistency Check Program consists of a system 2000 retrieval program, a main program with three subroutines and five functions. A description of the system 2000 retrieval program is given. The description of the main program has a structure diagram, a flow chart of the main program, followed by a write-up of the main program and each routine.

SFC	DISCHARGER	RECEIVER	CARNO	DATE	223	SHIFT	2	REC'D CAR	Q/C TYPE	17.25.71	PAGE	22
1	L-ATR	1	3575	1653		LCU		1720	FAC			
2	L-ATR	2	3524	1654		LCU		5285	FAC			
3	L-ATR	3	3701	1657		LCU		1720	FAC			
4	L-ATR	4	3733	1657		LCU		5352	FAC			
5	L-ATR	5	3786	1657		LCU		1720	FAC			
6	L-ATR	6	3805	1658		LCU		1726	FAC			
7	L-ATR	7	3833	1657		LCU		5435	FAC			
8	L-ATR	8	3836	1657		LCU		1720	FAC			
9	L-ATR	9	3867	1657		LCU		5352	FAC			
10	L-ATR	10	3905	1657		LCU		5467	FAC			
11	L-ATR	11	3912	1657		LCU		5447	FAC			
12	L-ATR	12	3930	1657		LCU		5455	FAC			
13	L-ATR	13	3941	1657		LCU		5328	FAC			
14	L-ATR	14	3974	1656		LCU		5449	FAC			
15	L-ATR	15	3984	1657		LCU		5440	FAC			
16	L-ATR	16	4143	1657		LCU		1725	FAC			
17	L-ATR	17	4167	1657		LCU		5435	FAC			
18	L-ATR	18	4167	1657		LCU		5435	FAC			
19	L-ATR	19	4167	1657		LCU		5435	FAC			
20	L-ATR	20	4252	1657		LCU		5440	FAC			
21	L-ATR	21	4253	1657		LCU		5435	FAC			
22	L-ATR	22	4300	1657		LCU		5447	FAC			
23	L-ATR	23	4307	1657		LCU		5440	FAC			
24	L-ATR	24	4313	1657		LCU		5285	FAC			
25	L-ATR	25	4348	1657		LCU		5314	FAC			
26	L-ATR	26	4358	1657		LCU		5449	FAC			
27	L-ATR	27	4401	1657		LCU		5445	FAC			
28	L-ATR	28	4466	1656		LCU		1721	FAC			
29	L-ATR	29	4557	1656		LCU		5358	FAC			
30	L-ATR	30	4565	1656		LCU		1720	FAC			
31	L-ATR	31	4642	1657		LCU		5285	FAC			
32	L-ATR	32	4754	1657		LCU		5440	FAC			
33	L-ATR	33	4762	1657		LCU		1726	FAC			
34	L-ATR	34	4768	1657		LCU		5435	FAC			
35	L-ATR	35	4772	1657		LCU		5314	FAC			
36	L-ATR	36	4778	1657		LCU		5447	FAC			
37	L-ATR	37	4781	1657		LCU		5447	FAC			
38	L-ATR	38	4815	1657		LCU		1726	FAC			
39	L-ATR	39	4815	1657		LCU		1726	FAC			
40	L-ATR	40	4845	1657		LCU		5447	FAC			
41	L-ATR	41	4991	1654		LCU		5316	FAC			

1 DISCHARGING CARRIER NON-SWITCHES
 1 RECEIVING CARRIER NON-SWITCHES
 2 DUPLICATE RECORDS
 3 INVALID CARRIER IDENTIFIERS

FIGURE 2.1. CONSISTENCY CHECK PROGRAM SAMPLE OUTPUT FOR THE ELEVATED CAUSEWAY
 CRANE ACTIVITIES FOR SHIFT 2, 11 August 1978. (FROM THE C1300 REPEATING GROUP
 OF THE LOTS DATA BASE.)

STC	CARGO ID	CARRIER ID	TYPE	RTF
1	1675	1663	LCU	1
2	1654	1654	LCU	2
3	1671	1657	LCU	3
4	1633	1663	LCU	4
5	1654	1643	LCU	5
6	1656	1657	LCU	6
7	1656	1656	LCU	7
8	1657	1657	LCU	8
9	1661	1661	LCU	9
10	1654	1654	LCU	10
11	1657	1663	LCU	11
12	1656	1656	LCU	12
13	1657	1657	LCU	13
14	1645	1645	LCU	14
15	166	166	LCU	15
16	1657	1657	LCU	16
17	1657	1657	LCU	17
18	1657	1657	LCU	18
19	1657	1657	LCU	19
20	1657	1657	LCU	20
21	1657	1657	LCU	21
22	1657	1657	LCU	22
23	1657	1657	LCU	23
24	1657	1657	LCU	24
25	1657	1657	LCU	25
26	1657	1657	LCU	26
27	1657	1657	LCU	27
28	1657	1657	LCU	28
29	1657	1657	LCU	29
30	1657	1657	LCU	30
31	1657	1657	LCU	31
32	1657	1657	LCU	32
33	1657	1657	LCU	33
34	1657	1657	LCU	34
35	1657	1657	LCU	35
36	1657	1657	LCU	36
37	1657	1657	LCU	37
38	1657	1657	LCU	38
39	1657	1657	LCU	39
40	1657	1657	LCU	40
41	1657	1657	LCU	41

2 DUPLICATE RECORDS
0 INVALID CARGO IDENTIFIERS

FIGURE 2.2. MATCHR1 SAMPLE OUTPUT FOR THE ELEVATED CAUSEWAY DISCHARGING LIGHTER ACTIVITIES FOR SHIFT 2, 11 AUGUST 1978. (FROM THE C1500 REPEATING GROUP OF THE LOTS DATA BASE.)

RECEIVING LAND CARRIER TABLE

SEC	CARRIER ID	TYPE	REF
1	1720	ENC	1
2	5286	ENC	2
3	1720	ENC	3
4	5358	ENC	4
5	5254	ENC	5
6	1720	ENC	6
7	1726	ENC	7
8	5445	ENC	8
9	1720	ENC	9
10	5286	ENC	10
11	5247	ENC	11
12	5247	ENC	12
13	5247	ENC	13
14	5247	ENC	14
15	5247	ENC	15
16	5247	ENC	16
17	5247	ENC	17
18	5247	ENC	18
19	5247	ENC	19
20	5247	ENC	20
21	5247	ENC	21
22	5247	ENC	22
23	5247	ENC	23
24	5247	ENC	24
25	5247	ENC	25
26	5247	ENC	26
27	5247	ENC	27
28	5247	ENC	28
29	5247	ENC	29
30	5247	ENC	30
31	5247	ENC	31
32	5247	ENC	32
33	5247	ENC	33
34	5247	ENC	34
35	5247	ENC	35
36	5247	ENC	36
37	5247	ENC	37
38	5247	ENC	38
39	5247	ENC	39
40	5247	ENC	40

NO MATCHES

2 NON-MATCHES
0 DUPLICATE RECORDS
0 INVALID CARGO IDENTIFIERS

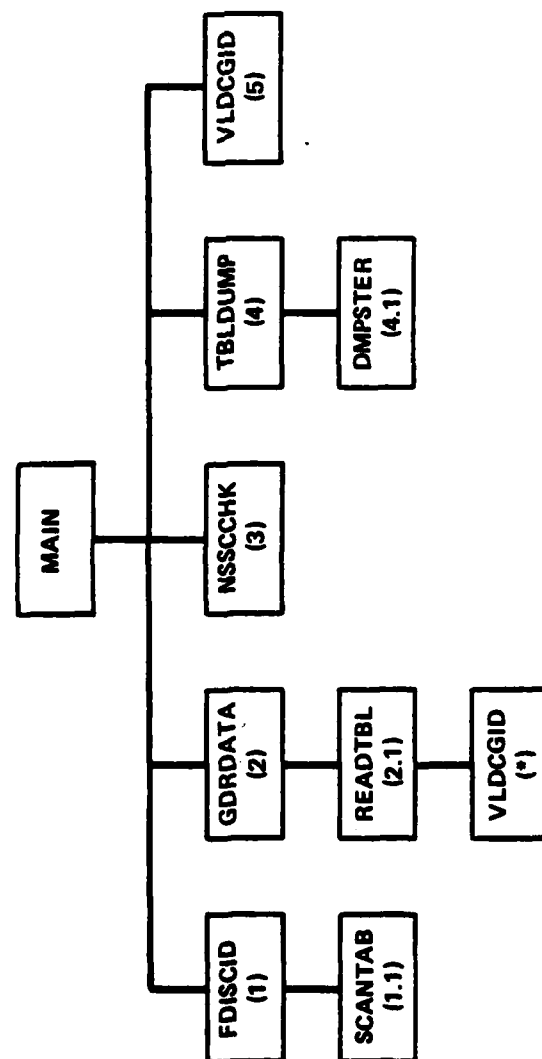
FIGURE 2.3. MATCHR1 SAMPLE OUTPUT FOR THE ELEVATED CAUSEWAY RECEIVING LAND CARRIER ACTIVITIES FOR SHIFT 2, 11 AUGUST 1978. (FROM THE C2000 REPEATING GROUP OF THE LOTS DATA BASE.)

SYSTEM 2000 RETRIEVALS FOR THE MATCHR1 FILES

Purpose

To create five data files for input to the MATCHR1 consistency check program.

<u>FILE NAME</u>	<u>TYPE</u>	<u>USE</u>	<u>DESCRIPTION</u>
CIBCA	BCD	Output	(1) Cargo ID (2) Discharging carrier ID (3) Discharging carrier type (4) Receiving carrier ID (5) Receiving carrier type (6) Facility ID (7) Date (8) Shift
CDVLT	BCD	Output	(1) Cargo discharged ID (2) Lighter ID (3) Lighter type (4) Facility ID (5) Date (6) Shift
CDVLC	BCD	Output	(1) LC Cargo discharged ID (2) Land carrier ID (3) Land carrier type (4) Facility ID (5) Date (6) Shift
CRVLT	BCD	Output	(1) LC Cargo received ID (2) Lighter ID (3) Lighter type (4) Facility ID (5) Date (6) Shift
CRVLC	BCD	Output	(1) LC Cargo received ID (2) Land carrier ID (3) Land carrier type (4) Facility ID (5) Date (6) Shift



* THIS FUNCTION IS ALSO CALLED BY MAIN.

FIGURE 2.4. MATCHR1 STRUCTURE

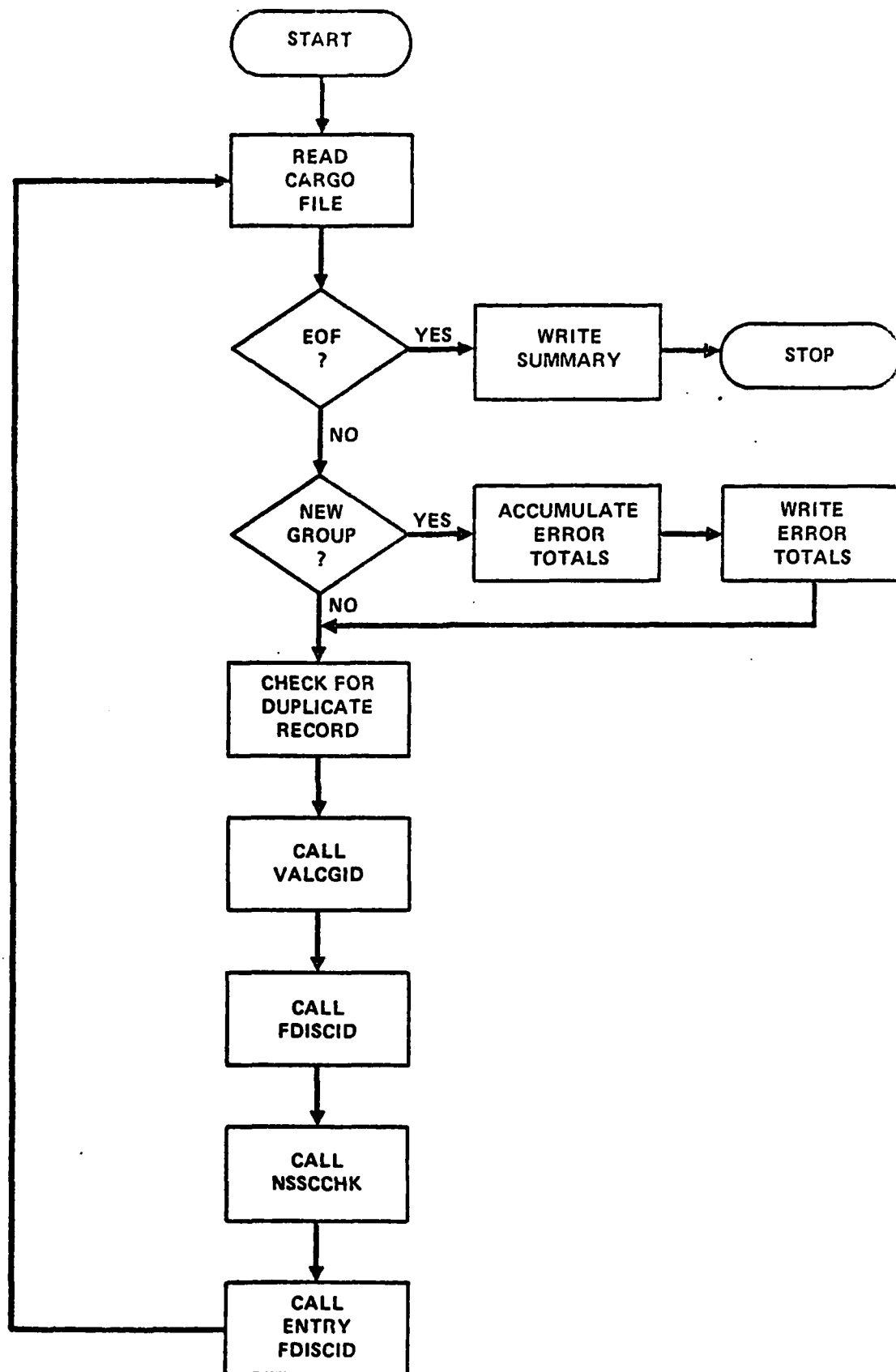


FIGURE 2.5. MAIN PROGRAM (MATCHR1) FLOW DIAGRAM

MAIN PROGRAM MATCHR1

Purpose

To perform a consistency check among selected components within the cargo activities, lighter activities, and land carrier repeating group structures of the LOTS SYSTEM 2000 data bases.

Program Statement

PROGRAM MATCHR1 (CIBCA, CDVLT, CDVLC, CRVLT, CRVLC, OUTPUT, CGIDTS)

<u>FILE NAME</u>	<u>TYPE</u>	<u>USE</u>	<u>DESCRIPTION</u>
CIBCA	BCD	Input	(1) Cargo ID (2) Discharging carrier ID (3) Discharging carrier type (4) Receiving carrier ID (5) Receiving carrier type (6) Facility ID (7) Date (8) Shift
CDVLT	BCD	Input	(1) Cargo discharged ID (2) Lighter ID (3) Lighter type (4) Facility ID (5) Date (6) Shift
CDVLC	BCD	Input	(1) LC Cargo discharged ID (2) Land carrier ID (3) Land carrier type (4) Facility ID (5) Date (6) Shift
CRVLT	BCD	Input	(1) LC Cargo received ID (2) Lighter ID (3) Lighter type (4) Facility ID (5) Date (6) Shift
CRVLC	BCD	Input	(1) LC Cargo received ID (2) Land carrier ID (3) Land carrier type (4) Facility ID (5) Date (6) Shift
Output	BCD	Output	The above five tables with inconsistencies marked are output. Invalid cargo ID's are indicated.
CGIDTS	BCD	Input	Valid cargo ID's

MAIN MATCHR1

FUNCTION FDISCID (1)

Purpose

To attempt to locate any entry in either of the discharging transporter tables whose cargo and carrier IDs matched those passed by the caller.

Calling Sequence

FDISCID (CGID, CRID, CIBCASQ)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
CGID	Integer	-	Input	Carrier ID
CRID	Integer	-	Input	Cargo ID
CIBCASQ	Integer	-	Input	Sequence number

COMMON BLOCKS

INPUT

OUTPUT

CDVLT	CDLCID, CDLTID, CDLTYP, CDLSEQ, NUMCDL
-------	---

CDVLC	CDCCID, CDCTID, CDCTYP, CDCSEQ, NUMCDC
-------	---

CRVLT	CRLCID, CRLTID, CRYTYP, CRLSEQ, NUMCRL
-------	---

CRVLC	CRCCID, CRCTID, CRCTYP, CRCSEQ, NUMCRL
-------	---

Functions Called

SCANTAB

MAIN MATCHR1

FUNCTION SCATTAB (1.1)

Purpose

To scan appropriate transporter table for a match with the record currently being processed.

Calling Sequence

SCANTAB (CARGTAB, CARRTAB, SQCIBCA, TABLIM, TABNAME, CARGID, CARRID, NSEQ, STRTSON)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
CARGTAB	Integer	TABLIM	Input	Cargo table
CARRTAB	Integer	TABLIM	Input	Carrier table
SQCIBCA	Integer	TABLIM	Input	Sequence number
TABLIM	Integer	-	Input	Dimension number
TABNAME	Integer	-	Input	Table name
CARGID	Integer	-	Input	Cargo ID
CARRID	Integer	-	Input	Carrier ID
NSEQ	Integer	-	Input	Sequence number of CIBCA Record
STRTSON	Integer	-	Output	Value of the first entry in the group currently being scanned.

Common Blocks

NONE

Subroutines Called

NONE

MAIN MATCHR1

SUBROUTINE GDRDATA (2)

Purpose

To obtain discharging/receiving transporter data from the appropriate files.

Calling Sequence

CALL GDRDATA

COMMON BLOCKS

TBLUNIT

CDVLT

CRVLT

CRVLC

INPUTS

NCIBCA, NCDVLT, NCDVLC,
NCRVLT, NCRVLC

OUTPUTS

CDLCID, CDLTID, CDLTYP,
CDLSEQ, NUMCDC

CRLCID, CRLTID, CRLTYP,
CRLSEQ, NUMCRL

CRCCID, CRCTID, CRCTYP,
CRCSEQ, NUMCRL

Functions Called

READTBL

MAIN MATCHR1

FUNCTION READTBL (2.1)

Purpose

To read values into the cargo, carrier, identifier and carrier-type arrays from the file whose unit number is passed as the value of the formal parameter "NUNIT"

Calling Sequence

READTBL (CARGTBL, CARRTBL, CRTPTBL, SEQTBL, NUNIT, FNAME, CALL1)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
CARGTBL	Integer	300	Output	Cargo ID
CARRTBL	Integer	300	Output	Carrier
CRTPTBL	Integer	300	Output	Carrier type
SEQTBL	Integer	300	Output	Sequence number
NUNIT	Integer	-	Input	Number of file
FNAME	Integer	-	Input	Name of file
CALL1	Logical	-	Input	First call flag

Common Blocks

NREPORT

NORMAN1

Inputs

NREPORT

CURFID, CURDATE, CURSHIFT

Outputs

Functions Called

VLDCCID

MAIN MATCHR1

FUNCTION NSSCCHK (3)

Purpose

To determine if NSSC code is unique.

Calling Sequence

NSSCCHK (IDENT, SEQ)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
IDENT	Integer	-	Input	NSSC CODE
SEQ	Integer	-	Input	Sequence number

Common Blocks

Input

Output

NREPORT
NSSCDUP

NREPORT

NSSCDUP

Subroutines Called

NONE

MAIN MATCHR1

SUBROUTINE TBLDUMP (4)

Purpose

To supervise the writing of the four transporter tables.

Calling Sequence

TBLDUMP

Common Blocks

Input

Output

DRTERRS

DLUNREP, DCUNREF, RLUNREF,
DLDUPL, DCDUPL, RLDUPL, RCDUPL,
DLINVAL, DCINVAL, RLINVAL, RCINVAL

CDVLT

CDLCID, CDLTID,
CDLTYP, CDLSEQ,
NUMCDL

CDVLC

CDCCID, CDCTID,
CDCTYP, CDCSEQ,
NUMCDC

CRVLT

CRLCID, CRLTID,
CRLTYP, CRLSEQ,
NUMCRL

CRVLD

CRCCID, CRCTID,
CRCTYP, CRCSEQ,
NUMCRC

Subroutines Called

DUMSTER

MAIN MATCHR1

SUBROUTINE DMPSTER (4.1)

Purpose

To write any one of four transporter tables.

Calling Sequence

DMPSTER (CARGTBL, CARRTBL, TYPETBL, SEQTBL, TBLSIZE, HEADING,NERARAY)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
CARTBL	Integer	TBLSIZE	Input	Cargo table
CARRTBL	Integer	TBLSIZE	Input	Carrier table
TYPETBL	Integer	TBLSIZE	Input	Type of transporter table
SEQTBL	Integer	TBLSIZE	Input	Sequence number of record in table
TBLSIZE	Integer	-	Input	Dimension number
HEADING	Real	-	Input	Table Heading
NERARAY	Integer	3	Output	(1) Number unreference records (2) Number duplicate records (3) Number invalid cargo IDs

Common Blocks

Input

Output

NORMAN1

FACILID, DATE, SHIFT

TITLE

CDATE,CTIME,DBNAME

NREPORT

NREPORT, PAGE

Subroutines Called

NONE

MAIN MATCHR1

FUNCTION VLDCGID (5)

Purpose

To determine if cargo ID is on the valid list of cargo IDs.

Calling Sequence

VLDCGID (IDENT)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
IDENT	Integer	-	Input	Cargo ID
<u>Common Blocks</u>		<u>Inputs</u>		<u>Outputs</u>
CGTUNIT		CTUNIT		
NREPORT		NREPORT		

Subroutines Called

NONE

COMMON BLOCK CDVLC

<u>Name</u>	<u>Type</u>	<u>Dimension</u>	<u>Description</u>
CDCCID	Integer	300	Discharging Land Carrier Cargo ID
CDCTID	Integer	300	Discharging Land Carrier ID
CDCTYP	Integer	300	Discharging Land Carrier Type
CDCSEQ	Integer	300	Discharging Land Carrier Sequence Number
NUMCDC	Integer	---	Number of Discharging Land Carrier Records

COMMON BLOCK COVLT

CDLCID	Integer	300	Discharging Lighter Cargo ID
CDLTID	Integer	300	Discharging Lighter ID
CDLTYP	Integer	300	Discharging Lighter Type
CDLSEQ	Integer	300	Discharging Lighter Sequence Number
NUMCDL	Integer	---	Number of Discharging Lighter Records

COMMON BLOCK CGTUNIT

NCGUNIT	Integer	---	I/O Unit Number
---------	---------	-----	-----------------

COMMON BLOCK CRVLC

CRCCID	Integer	300	Receiving Land Carrier Cargo ID
CRCTID	Integer	300	Receiving Land Carrier ID
CRCTYP	Integer	300	Receiving Land Carrier Type
CRCSEQ	Integer	300	Receiving Land Carrier Sequence Number
NUMCRC	Integer	---	Number of Receiving Land Carrier Records.

COMMON BLOCK CRVLT

CRLCID	Integer	300	Receiving Lighter Cargo ID
CRLTID	Integer	300	Receiving Lighter ID
CRLTYP	Integer	300	Receiving Lighter Type
CRLSEQ	Integer	300	Receiving Lighter Sequence Number
NUMCRL	Integer	---	Number of Receiving Lighter Records

COMMON BLOCK DRTERRS

DLUNRER	Integer	---	Number of Discharging Lighter Non-Matches Records
---------	---------	-----	--

COMMON BLOCK DRTERRS (CONT.)

<u>Name</u>	<u>Type</u>	<u>Dimension</u>	<u>Description</u>
DCUNRER	Integer	---	Number of Discharging Land Carrier Non-Matches
RLUNREF	Integer	---	Number of Receiving Lighter Non-Matches
RCUNREF	Integer	---	Number of Receiving Land Carrier Non-Matches
DL DUP	Integer	---	Number of Discharging Lighter Duplicates
DC DUP	Integer	---	Number of Discharging Land Carrier Duplicates
RL DUPL	Integer	---	Number of Receiving Lighter Duplicates
RC DUP	Integer	---	Number of Receiving Land Carrier Duplicates
DLINVAL	Integer	---	Number of Discharging Lighter Invalid Cargo IDs
DCINVAL	Integer	---	Number of Discharging Land Carrier Invalid Cargo IDs
RLINVAL	Integer	---	Number of Invalid Receiving Lighter Invalid Cargo IDs
RCINVAL	Integer	---	Number of Receiving Land Carrier Invalid Cargo IDs.
NFLRECS	Integer	---	Total Discharging Lighter Records
NDCRECS	Integer	---	Total Discharging Land Carrier Records
NRLRECS	Integer	---	Total Receiving Lighter Records
NRCRECS	Integer	---	Total Receiving Land Carrier Records

COMMON BLOCK NORMAN1

PRVFID	Integer	---	Facility ID on Previous Record
PRVDATE	Integer	---	Date on Previous Record
PRVSHFT	Integer	---	Shift on Previous Record

COMMON BLOCK NREPORT

NREPORT	Integer	---	I/O Unit Number
---------	---------	-----	-----------------

COMMON BLOCK NSSCDUP

NSSCDUP	Integer	---	Number of NSSC Duplicate Codes
---------	---------	-----	--------------------------------

COMMON BLOCK PAGENUM

<u>Name</u>	<u>Type</u>	<u>Dimension</u>	<u>Description</u>
NPAGE	Integer	---	Current Output Page Number

COMMON BLOCK TBLUNIT

NCIBCA	Integer	---	I/O Unit Number
NCDULT	Integer	---	I/O Unit Number
NCDVLC	Integer	---	I/O Unit Number
NCRVLT	Integer	---	I/O Unit Number
NCRVLC	Integer	---	I/O Unit Number

COMMON BLOCK TITLE

CDATE	Real	---	Current Date
CTIME	Real	---	Current Time
DBNAME	Real	---	Data Base Name

III. LIGHTER CYCLE TIMES PROGRAM

INTRODUCTION

The Lighter Cycle Times Program traces lighters carrying containers between ship and shore facilities. The following information is given for each lighter at the ship:

- 1) Ship facility
- 2) Date
- 3) Shift
- 4) Lighter type
- 5) Lighter ID
- 6) Lighter cycle
- 7) Lighter position at the ship
- 8) Forward or retrograde movement
- 9) Lighter ready at the ship time (LDRY)
- 10) Lighter ready to depart the ship time (LRDP)
- 11) Lighter at the ship time (LRDP-LRDY)

Data for the shore part of the record are:

- 1) Shore facility
- 2) Lighter arrived at the shore time (LARR)
- 3) Lighter unclear way time (LARR at the shore-LRDP at the ship)
- 4) Lighter in position at the shore time (LPSN)
- 5) Lighter position time (LPSN-LARR)
- 6) Lighter ready at the shore time (LRDY)
- 7) Lighter ready time (LRDY-LPSN)
- 8) Lighter ready to depart the shore (LRDP)
- 9) Lighter at crane time (LRDP-LRDY)
- 10) Lighter succession time (LRDY-previous LRDP)
- 11) Previous lighter type

The mean and standard deviation are computed and presented in the output for all time differences by ship facility, date and shift.

A sample output is given in Figure 3.1.

PROGRAM DESCRIPTION

The Lighter Cycle Times Program consists of two System 2000 retrieval programs, five main programs containing 32 subprograms, four sorts and one merge. A flow chart of the steps in this program is given in Figure 3.2. A description of each program is given starting with the two System 2000 retrieval programs and followed by the five main programs. The description of each main program contains a structure diagram, a flow chart of the main program, and a write-up of the main program and each subroutine.



FIGURE 3.1. LIGHTER CYCLE TIMES PROGRAM SAMPLE OUTPUT

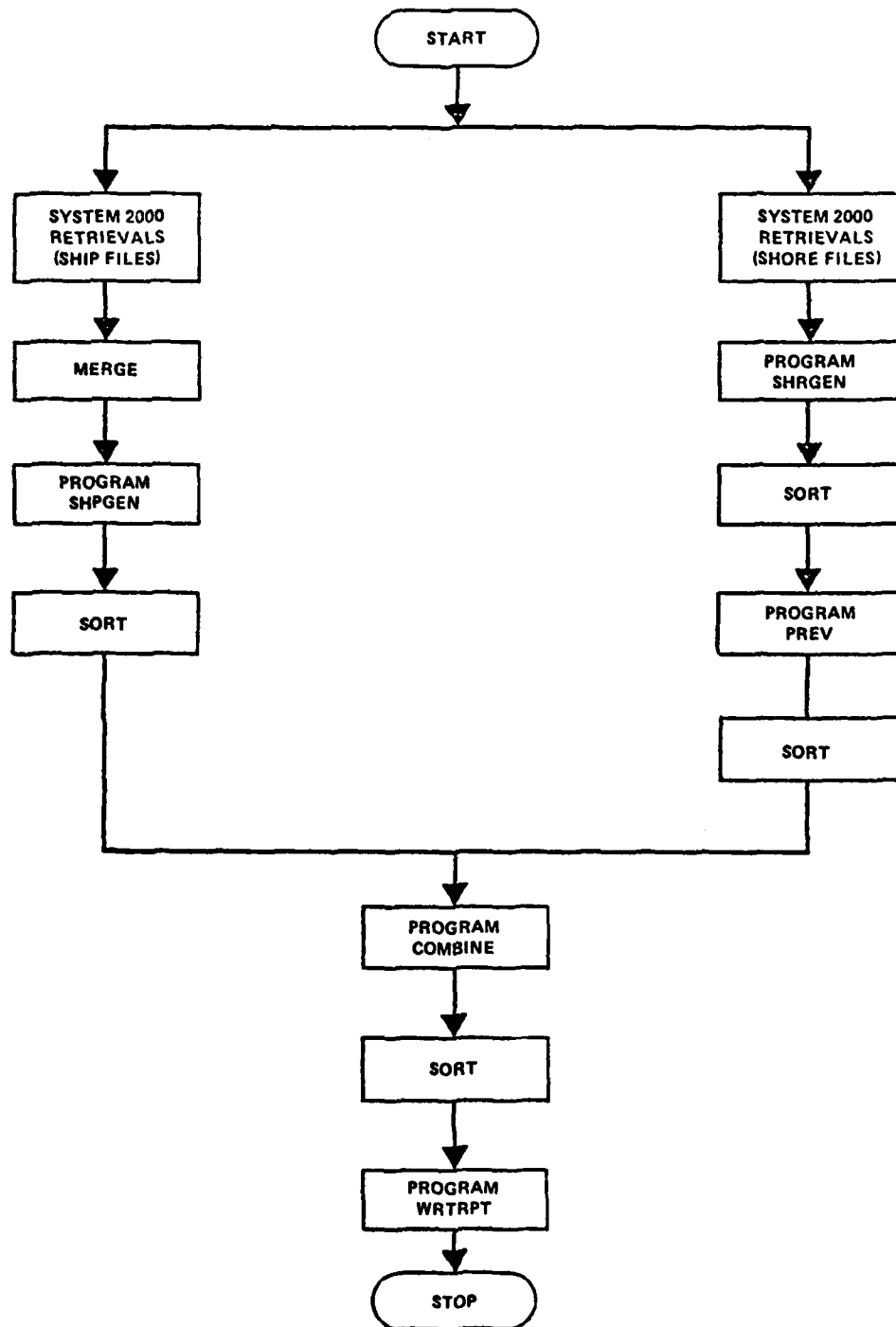


FIGURE 3.2. FLOW CHART OF THE STEPS IN THE LIGHTER CYCLE TIMES PROGRAM

SYSTEM 2000 RETRIEVALS FOR THE SHIP FILES

Purpose

To retrieve lighter data at the ship from the LOTS System 2000 data base and to store the data on four files.

<u>FILE NAME</u>	<u>TYPE</u>	<u>USE</u>	<u>DESCRIPTION</u>
LRDY1	BCD	Output	A LRDY1 record contains: (1) Facility ID (COD or TCDF) (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Lighter event time (LRDY)
LRDP1	BCD	Output	A LRDP1 record contains: (1) Facility ID (COD or TCDF) (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter event time (LRDP)
F1	BCD	Output	A F1 record indicates forward lighter movement. The record contains: (1) Facility ID (COD or TCDF) (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Forward (an F is added to this file using the system editor).
R1	BCD	Output	A R1 record indicates retrograde lighter movement. The record contains: (1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Retrograde (an R is added to this record using the system editor).

SYSTEM 2000 RETRIEVALS FOR THE SHORE FILES

Purpose

To retrieve lighter data at the shore from the LOTS System 2000 data base and to store the data on four data bases.

<u>FILE NAME</u>	<u>TYPE</u>	<u>USE</u>	<u>DESCRIPTION</u>
LARR2	BCD	Output	A LARR2 record contains: (1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter event time (LARR)
LPSN2	BCD	Output	A LPSN2 record contains: (1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter event time (LPSN)
LRDY2	BCD	Output	A LRDY2 record contains: (1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter event time (LRDY)
LRDP2	BCD	Output	A LRDP2 record contains: (1) Facility date (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter event time (LRDP)

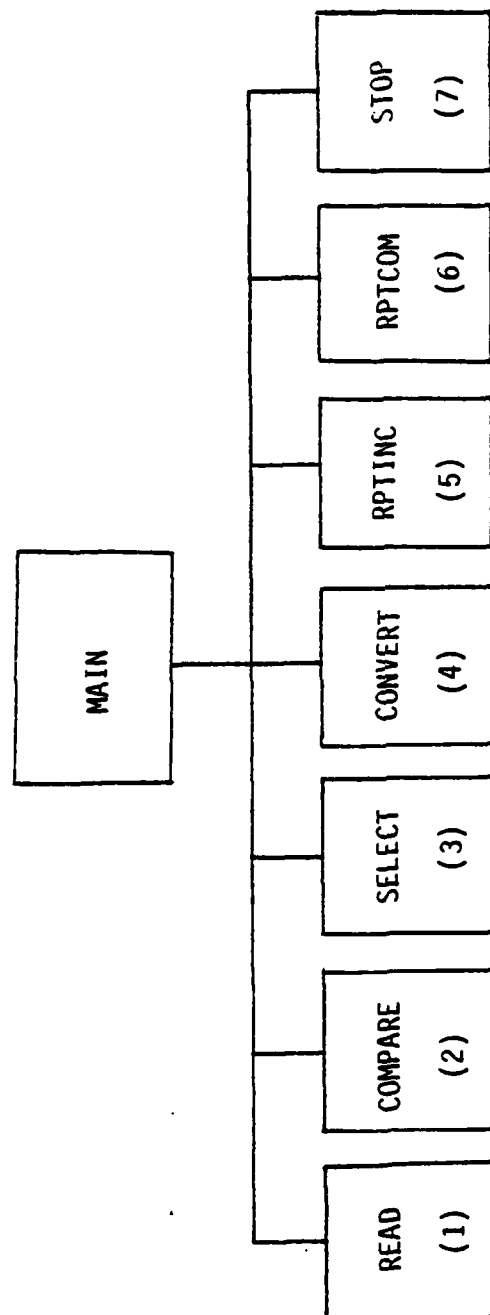


FIGURE 3.3. SHPGEN STRUCTURE

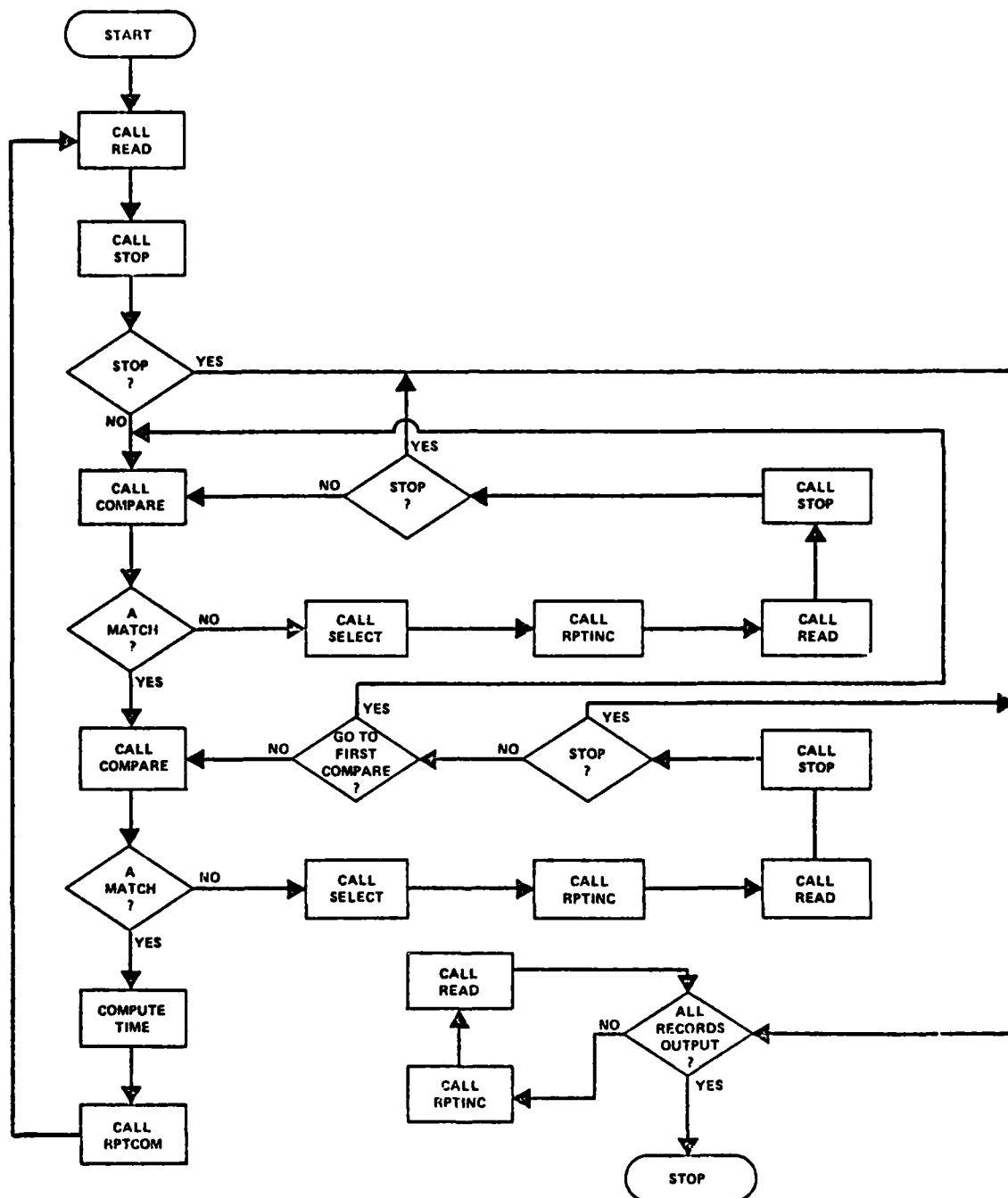


FIGURE 3.4. MAIN PROGRAM (SHPGEN) FLOW DIAGRAM

MAIN PROGRAM SHPGEN

Purpose

To combine three records, contained in the System 2000 ship files, into one record.

Program Statement

PROGRAM SHPGEN (LRDY1, LRDP1, FR1, OUTI1, OUTC1, OUTPUT, OUTBN1)

<u>FILE NAME</u>	<u>TYPE</u>	<u>USE</u>	<u>DESCRIPTION</u>
LRDY1	BCD	Input	A LRDY1 record contains: (1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Lighter event time (LRDY)
LRDP1	BCD	Input	A LRDP1 records contains: (1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter event time (LRDP)
NFR	BCD	Input	A NFR record contains: (1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Foreward or retrograde (F/R)
OUTI1	BCD	Output	Any LRDY1, LRDP2 or NFR record that doesn't have matching records in the other two files.

<u>FILE NAME</u>	<u>TYPE</u>	<u>USE</u>	<u>DESCRIPTION</u>
OUTC1	BCD	Output	A OUTC1 record contains: (1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Forward or retrograde (F/R) (9) Lighter event time (LRDY) (10) Lighter event time (LRDP) (11) Lighter Ship time (LRDP-LRDY) (12) Minutes elapsed from the beginning of the test.
OUTPUT	BCD	Output	Contains system messages, if any.
OUTBN1	Binary	Output	Contains the same record as OUTC1.

MAIN SHPGEN

SUBROUTINE READ (1)

Purpose

To read one record from one, two or three input files depending on the value of KODE.

Calling Sequence

CALL READ (KODE, NLRDY, NLRDP, NFR, NLGTPS, NPAM, NFIRST, NEND)

<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
KODE	Integer	3	Input	(1) Flag to read a file 1 record. (2) Flag to read a file 2 record. (3) Flag to read a file 3 record.
NFIRST	Logical	-	Input	Equals false on first call and true on additional calls.
NLRDY	Integer	6	Output	First six parameters of input file 1 record, i.e., (1) Facility ID (2) Date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle
NLRDP	Integer	6	Output	First six parameters of input file 2 record, i.e., (1) Facility ID (2) Date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle
NFR	Integer	6	Output	First six parameters of input file 3 record, i.e., (1) Facility ID (2) Date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle
NLGTPS	Integer	-	Output	Seventh parameter of input file 1 record; i.e., lighter position.
NPAM	Integer	3	Output	Last parameter of each of the three input file records, i.e., (1) File 1 record-lighter event time (LRDY) (2) File 2 record-lighter event time (LRDY) (3) File 3 record-forward or retro-grade movement (F/R)
NEND	Integer	3	Output	Equals EOF when end-of-file is reached.

Common Blocks

<u>Block Name</u>	<u>Inputs</u>	<u>Output</u>
TBLUNIT	NUNIT1, NUNIT2, NUNIT3, NUNIT4, NUNIT5, NUNIT6, NUNIT7	None

Subroutines Called

None

MAIN SHPGEN

SUBROUTINE COMPARE (2)

Purpose

To determine if two records match on six match keys. If there is no match, it returns to the main program the first match key where the match failed.

Calling Sequence

CALL COMPARE (NTEST1, NTEST2, MATCH, NFAIL1, NFAIL2)

<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
NTEST1	Integer	6	Input	Contains the lighter ready record (six match keys).
NTEST2	Integer	6	Input	Contains the lighter ready to depart or forward/retrograde record (six match keys).
MATCH	Logical	-	Input	MATCH equals true if a match was found. MATCH equals false if the match failed.
NFAIL1	Integer	-	Output	Contains the first lighter ready match key where the match failed.
NFAIL2	Integer	-	Output	Contains the first lighter ready or the first forward/retrograde match key where the match failed.

Common Blocks

<u>Block Name</u>	<u>Inputs</u>	<u>Output</u>
None	None	None

Subroutines Called

None

MAIN SHPGEN

SUBROUTINE SELECT (3)

Purpose

To determine which record to read.

Calling Sequence

CALL SELECT (KEY, NFAIL1, NFAIL2, KODE)

<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
KEY	Integer	-	Input	When KEY equals 1, the match between a lighter ready record and a lighter ready to depart record failed. When KEY equals 2, the match between a lighter ready record and a forward/retrograde record failed.
NFAIL1	Integer	-	Input	Contains the first lighter ready match key where the match failed.
NFAIL2	Integer	-	Input	Contains the first lighter ready or the first foreward/retrograde match key where the match failed.
KODE	Integer	3	Output	If match failed: (1) Flag to indicate an incomplete file 1 record. (2) Flag to indicate an incomplete file 2 record. (3) Flag to indicate an incomplete file 3 record.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
None	None	None

Subroutines Called

None

MAIN SHPGEN

SUBROUTINE CONVERT (4)

Purpose

Converts day, shift, hours, and minutes to minutes from the beginning of the test.

Calling Sequence

CALL CONVERT (NDAY, NSHIFT, NHM, MINS)

<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
NDAY	Integer	-	Input	Julian date.
NSHIFT	Integer	-	Input	Shift 1 or shift 2.
NHM	Integer	-	Input	Time of day in hours and minutes (military time).
MINS	Integer	-	Output	Minutes elapsed from the beginning of the test.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
None	None	None

Subroutines Called

None

MAIN SHPGEN

SUBROUTINE RPTINC (5)

Purpose

To write an incomplete record.

Calling Sequence

CALL RPTINC (KODE, NLRDY, NLRDP, NFR, NLGTPS, NPAM, NFINC)

<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
KODE	Integer	3	Input	(1) Flag to write a file 1 record. (2) Flag to write a file 2 record. (3) Flag to write a file 3 record.
NLRDY	Integer	6	Input	First six parameters of input file 1 record, i.e., (1) Facility ID (2) Date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle
NLRDP	Integer	6	Input	First six parameters of input file 2 record, i.e., (1) Facility ID (2) Date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle
NFR	Integer	6	Input	First six parameters of input file 3 record, i.e., (1) Facility ID (2) Date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle
NLGTPS	Integer	-	Input	Seventh parameter of input file 1 record, i.e., lighter position.
NPAM	Integer	3	Input	Last parameter of each of the three input file records, i.e., (1) File 1 record-lighter event time (LRDY) (2) File 2 record-lighter event time (LRDY) (3) File 3 record-forward or retrograde movement (F/R)
NFINC	Logical	-	Input	Equals false on first call and true on additional calls. When NFINC equals false, automatically prints complete record file title.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
TBLUNIT	NUNIT4	None

Subroutines Called

None

MAIN SHPGEN

SUBROUTINE RPTCOM (6)

Purpose

To write a complete record.

Calling Sequence

CALL RPTCOM (NLRDY, NLGTPS, NPAM, NFCOM, LSHPT, MINS)

<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
NLRDY	Integer	6	Input	First six parameters of input file 1 record, i.e., (1) Facility ID (2) Date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle
NLGTPS	Integer	-	Input	Seventh parameter of input file 1 record, i.e., lighter position.
NPAM	Integer	3	Input	Last parameter of each of the three input file records, i.e., (1) File 1 record-lighter event time (LRDY) (2) File 2 record-lighter event time (LRDY) (3) File 3 record-forward or retrograde movement (F/R).
NFCOM	Logical	-	Input	Equals false on first call and true on additional calls. When NFCOM equals false, automatically prints complete record file title.
LSHPT	Integer	-	Input	Lighter at ship time (LRDP-LRDY).
MINS	Integer	-	Input	Minutes elapsed from the beginning of the test.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
TBLUNIT	NUNIT5, NUNIT7	None

Subroutines Called

None

MAIN SHPGEN

SUBROUTINE STOP (7)

Purpose

To determine if end-of-file is in any or in all files. Sets KODE to write incomplete records.

Calling Sequence

CALL STOP (NEND, NSTOP, NSTOP1, KODE)

<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
NEND	Integer	3	Input	Equals EOF when end of file is reached.
NSTOP	Integer	-	Output	When NSTOP is false, all files contain data. When NSTOP is true, at least one file is out of data.
NSTOP1	Logical	-	Output	When NSTOP1 is true, all files are out of data.
KODE	Integer	3	Output	(1) Flag to read a file 1 record. (2) Flag to read a file 2 record. (3) Flag to read a file 3 record.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
None	None	None

Subroutines Called

None

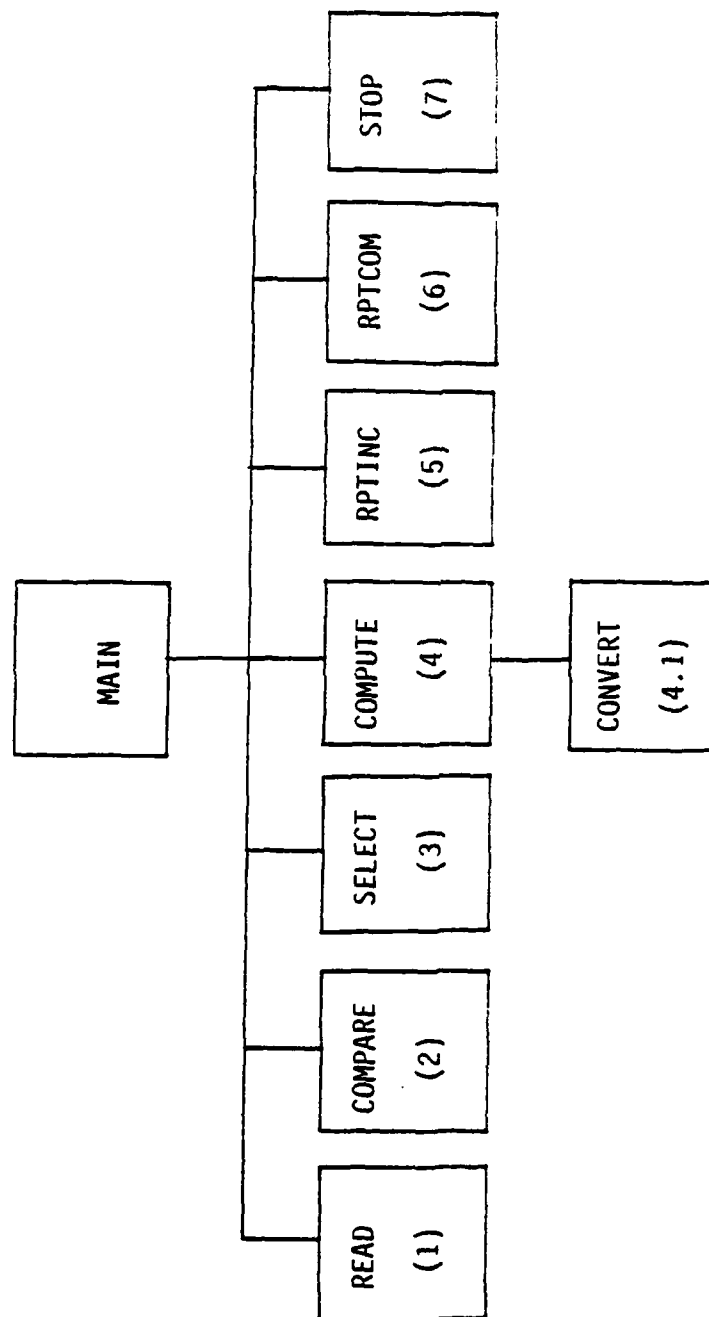


FIGURE 3.5. SHRGEN STRUCTURE

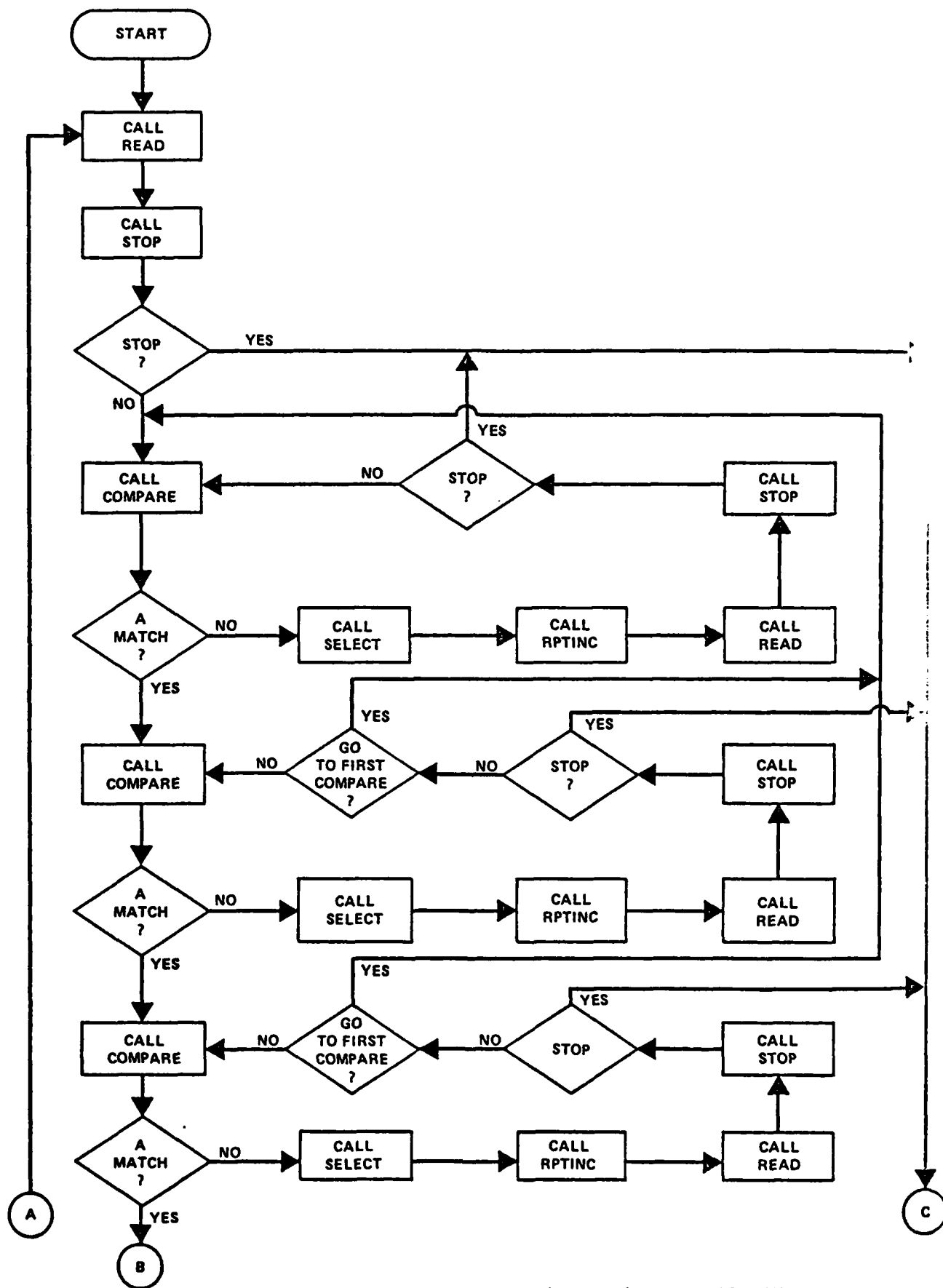


FIGURE 3.6. MAIN PROGRAM (SHRGEN) FLOW DIAGRAM

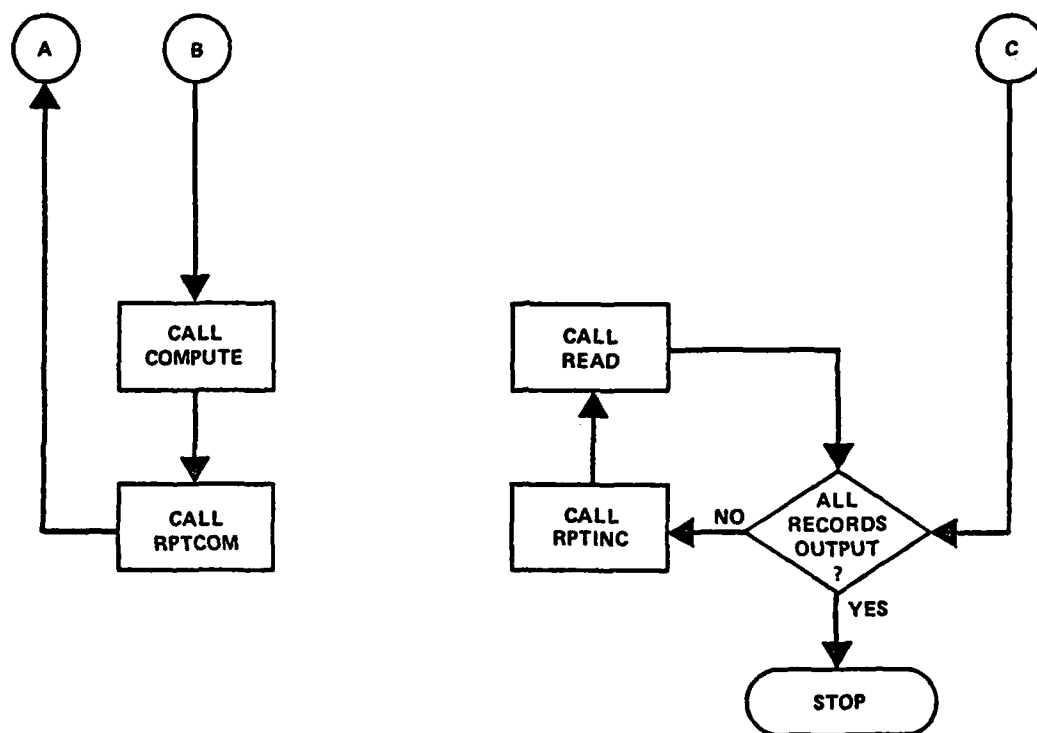


FIGURE 3.7. (CONT'D) MAIN PROGRAM (SHRGEN) FLOW DIAGRAM

MAIN PROGRAM SHRGEN

Purpose

To combine the four records, contained in the system 2000 shore files, into one record.

Program Statement

PROGRAM SHRGEN (LARR2, LPSN2, LRDY2, LRDP2, OUTI2, OUTC2, OUTBN2, OUTPUT)

<u>FILE NAME</u>	<u>TYPE</u>	<u>USE</u>	<u>DESCRIPTION</u>
LARR2	BCD	Input	A LARR2 record contains: (1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter event time (LARR)
LPSN2	BCD	Input	A LPSN2 record contains: (1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter event time (LRDP)
LRDY2	BCD	Input	A LRDY2 record contains: (1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter event time (LRDY)
LRDP2	BCD	Input	A LRDP2 record contains: (1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter event time (LRDP2)
OUTI2	BCD	Output	Any LARR2, LPSN2, LRDY2 or LRDP2 record that doesn't have a matching records in the other three files.

<u>FILE NAME</u>	<u>TYPE</u>	<u>USE</u>	<u>DESCRIPTION</u>
OUTC2	BCD	Output	A OUTC2 record contains: (1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter event time (LARR) (8) Lighter event time (LPSN) (9) Lighter event time (LRDY) (10) Lighter event time (LRDP) (11) Lighter in position time (LPSN-LARR) (12) Lighter ready time (LRDY-LPSN) (13) Lighter at crane time (LRDP-LPSN) (14) Minutes elapsed from the beginning of the test.
OUTBN2	Binary	Output	Contains the same record as OUTC2.
OUTPUT	BCD	Output	Contains systems messages, if any.

MAIN SHRGEN

SUBROUTINE READ (1)

Purpose

To read one record from any of four files depending on the value of
KODE.

Calling Sequence

CALL READ (KODE, NLARR, NLPSN, NLRDY, NLRDP, NFIRST, NEND)

<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
KODE	Integer	4	Input	(1) Flag to read a file 1 record. (2) Flag to read a file 2 record. (3) Flag to read a file 3 record. (4) Flag to read a file 4 record.
NLARR	Integer	7	Output	First seven parameters of input file 1 record, i.e., (1) Facility ID (2) Date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter event time (LARR)
NLPSN	Integer	7	Output	First seven parameters of input file 2 record, i.e., (1) Facility ID (2) Date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter event time (LPSN).
NLRDY	Integer	7	Output	First seven parameters of input file 3 record, i.e., (1) Facility ID (2) Date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter event time (LRDY)
NLRDP	Integer	7	Output	First seven parameters of input file 4 record, i.e., (1) Facility ID (2) Date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter event time (LRDP)

NFIRST	Logical	-	Output	Equals false on first call and true on additional calls.
NEND	Integer	4	Output	Equals EOF when end-of-file is reached.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
TBLUNIT	NUNIT1, NUNIT2, NUNIT3, NUNIT4, NUNIT5, NUNIT6, NUNIT7, NUNIT8	None

Subroutines Called

None

MAIN SHRGEN

SUBROUTINE COMPARE (2)

Purpose

To determine if two records match on six match keys. If there is no match, it returns to the main program the first match key where the match failed.

Calling Sequence

CALL COMPARE (NTEST1, NTEST2, MATCH, NFAIL1, NFAIL2)

<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
NTEST1	Integer	7	Input	Contains the lighter arrival record (six match keys).
NTEST2	Integer	7	Input	Contains the lighter in position, the lighter ready, or the lighter ready to depart record (six match keys).
MATCH	Logical	-	Output	MATCH equals true if a match was found. MATCH equals false if the match failed.
NFAIL1	Integer	-	Output	Contains the first lighter arrival match key where the match failed.
NFAIL2	Integer	-	Output	Contains the first lighter in position, lighter ready, or lighter ready to depart match key where the match failed.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
None	None	None

Subroutines Called

None

MAIN SHRGEN

SUBROUTINE SELECT (3)

Purpose

To determine which record to read.

Calling Sequence

CALL SELECT (KEY, NFAIL1, NFAIL2, KODE)

<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
KEY	Integer	-	Input	When KEY equals 1, the match between a lighter arrival record and a lighter in position record failed. When KEY equals 2, the match between a lighter arrival record and a lighter ready record failed. When KEY equals 3, the match between a lighter arrival record and a lighter ready to depart record failed.
NFAIL1	Integer	-	Input	Contains the first lighter arrival match key where the match failed.
NFAIL2	Integer	-	Input	Contains the first lighter in position match key where the match failed.
KODE	Integer	4		If match failed: (1) Flag to indicate an incomplete file 1 record. (2) Flag to indicate an incomplete file 2 record. (3) Flag to indicate an incomplete file 3 record. (4) Flag to indicate an incomplete file 4 record.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
None	None	None

Subroutines Called

None

MAIN SHRGEN

SUBROUTINE COMPUTE (4)

Purpose

To compute lighter in position time, lighter ready time, lighter at crane time, and time elapsed from the beginning of the test.

Calling Sequence

CALL COMPUTE (NLARR, NLPSN, NLRDY, NLRDP, NTIME)

<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
NLARR	Integer	7	Input	First seven parameters of input file 1 record, i.e., (1) Facility ID (2) Date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter event time (LARR)
NLPSN	Integer	7	Input	First seven parameters of input file 2 record, i.e., (1) Facility ID (2) Date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter event time (LPSN)
NLRDY	Integer	7	Input	First seven parameters of input file 3 record, i.e., (1) Facility ID (2) Date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter event time (LRDY)
NLRDP	Integer	7	Input	First seven parameters of input file 4 record, i.e., (1) Facility ID (2) Date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter event time (LRDP)
NTIME	Integer	4	Output	(1) Equals lighter in position time (LPSN-LARR) (2) Equals lighter ready time (LRDY-LPSN)

(3) Equals lighter at crane time
(LRDP-LRDY)
(4) Equals time from the beginning of
the test (LRDY).

Common Blocks

<u>Block Name</u>	<u>Input</u>
None	None

<u>Output</u>
None

Subroutines Called

Convert

MAIN SHRGEN

SUBROUTINE CONVERT (4.1)

Purpose

Converts days, shifts, hours, and minutes to minutes from the beginning of the test.

CALL CONVERT (NARRAY, MINS)

<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
NARRAY	Integer	7	Input	(2) Julian date (3) Shift (7) Time of day in hours and minutes (military time).
MINS	Integer	-	Output	Minutes elapsed from the beginning of the test.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
None	None	None

Subroutines Called

None

MAIN SHRGEN

SUBROUTINE RPTINC (5)

Purpose

To write an incomplete record.

Calling Sequence

CALL RPTINC (KODE, NLARR, NLPSN, NLRDY, NLRDP, NFINC)

<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
KODE	Integer	4	Input	(1) Flag to write a file 1 record. (2) Flag to write a file 2 record. (3) Flag to write a file 3 record. (4) Flag to write a file 4 record.
NLARR	Integer	7	Input	First seven parameters of input file 1 record, i.e., (1) Facility ID (2) Date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter event time (LARR)
NLPSN	Integer	7	Input	First seven parameters of input file 2 record, i.e., (1) Facility ID (2) Date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter event time (LPSN)
NLRDY	Integer	7	Input	First seven parameters of input file 3 record, i.e., (1) Facility ID (2) Date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter event time (LRDY)
NLRDP	Integer	7	Input	First seven parameters of input file 4 record, i.e., (1) Facility ID (2) Date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter event time (LRDP)

NFINC	Logical	-	Input	Equals false on first call and true on additional calls. When NFINC equals false, automatically prints complete record file title.
-------	---------	---	-------	--

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
TBLUNIT	NUNIT1, NUNIT2, NUNIT3, NUNIT4, NUNIT5, NUNIT6, NUNIT7, NUNIT8	None

Subroutines Called

None

MAIN SHRGEN

SUBROUTINE RPTCOM (6)

Purpose

To write a complete record.

Calling Sequence

CALL RPTCOM (NLARR, NLPSN, NLRDY, NLRDP, NTIME, NFCOM)

<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
NLARR	Integer	7	Input	First seven parameters of input file 1 record, i.e., (1) Facility ID (2) Date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter event time (LARR)
NLPSM	Integer	7	Input	First seven parameters of input file 2 record, i.e., (1) Facility ID (2) Date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter event time (LPSN)
NLRDY	Integer	7	Input	First seven parameters of input file 3 record, i.e., (1) Facility ID (2) Date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter event time (LRDY)
NLRDP	Integer	7	Input	First seven parameters of input file 4 record, i.e., (1) Facility ID (2) Date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter event time (LRDP)
NTIME	Integer	4	Input	(1) NTIME1 equals lighter in position time (LPSN-LARR). (2) NTIME2 equals lighter ready time (LRDY-LPSN).

NFCOM	Logical	-	Input	<p>(3) NTIME3 equals lighter at crane time (LRDP-LPSN).</p> <p>(4) NTIME4 equals time from the beginning of the test (LRDY).</p> <p>Equals false on first call and true on additional calls. When NFCOM equals false, automatically prints complete record file.</p>
-------	---------	---	-------	--

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
TBLUNIT	NUNIT1, NUNIT2, NUNIT3, NUNIT4, NUNIT5, NUNIT6, NUNIT7, NUNIT8	None

Subroutines Called

None

MAIN SHRGEN

SUBROUTINE STOP (7)

Purpose

To determine if end-of-file is in any or in all files. Sets KODE to write incomplete records.

Calling Sequence

CALL STOP (NEND, NSTOP, NSTOP1, KODE)

<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
NEND	Integer	4	Input	Equals EOF when end-of-file is reached.
NSTOP	Logical	-	Output	When NSTOP is false, all files contain data. When NSTOP is true at least one file is out of data.
NSTOP1	Logical	-	Output	When NSTOP1 is true, all files are out of data.
KODE	Integer	4	Output	(1) Flag to read a file 1 record. (2) Flag to read a file 2 record. (3) Flag to read a file 3 record. (4) Flag to read a file 4 record.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
None	None	None

Subroutines Called

None

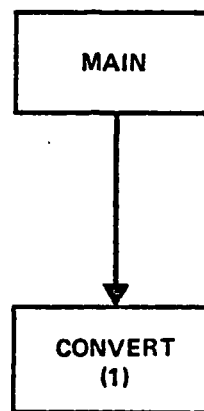


FIGURE 3.8. PREV STRUCTURE

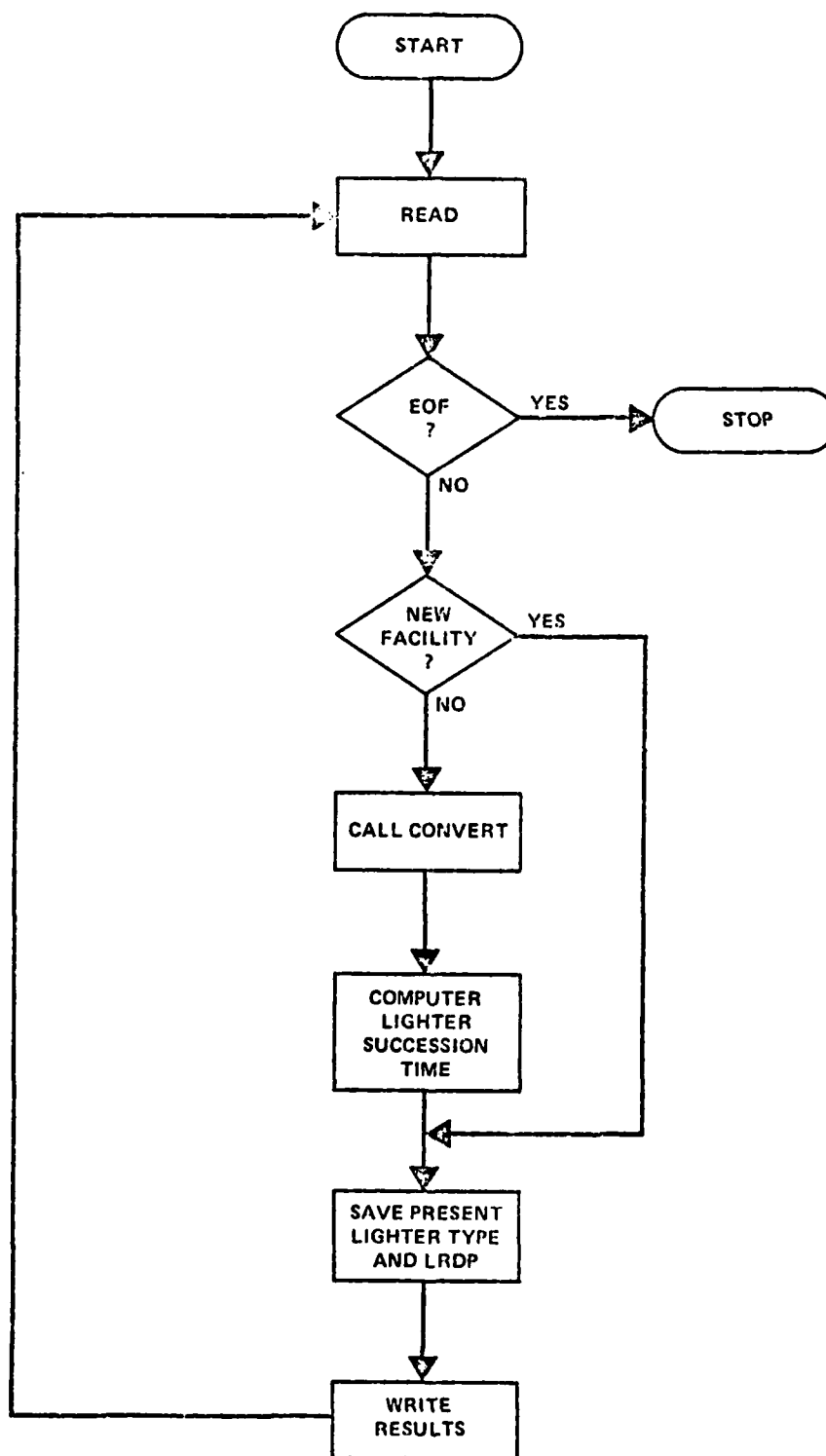


FIGURE 3.9. MAIN PROGRAM (PREV) FLOW DIAGRAM

MAIN PROGRAM PREV

Purpose

To save previous lighter type and previous LRDP (SHORE) on present record. To compute lighter succession time at the shore.

Program Statement

PROGRAM PREV (TAPE1, OUT, OUTPUT)

<u>FILE NAME</u>	<u>TYPE</u>	<u>USE</u>	<u>DESCRIPTION</u>
TAPE1	Binary	Input	<p>A TAPE1 record contains:</p> <ol style="list-style-type: none"> (1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter event time (LARR) (8) Lighter event time (LPSN) (9) Lighter event time (LRDY) (10) Lighter event time (LRDP) (11) Lighter in position time (LPSN-LARR) (12) Lighter ready time (LRDY-LPSN) (13) Lighter at crane time (LRDP-LPSN) (14) Minutes elapsed from the beginning of the test.
OUT	Binary	Output	<ol style="list-style-type: none"> (1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter event time (LARR) (8) Lighter event time (LPSN) (9) Lighter event time (LRDY) (10) Lighter event time (LRDP) (11) Lighter in position time (LPSN-LARR) (12) Lighter ready time (LRDY-LPSN) (13) Lighter at crane time (LRDP-LPSN) (14) Minutes elapsed from the beginning of the test. (15) Lighter succession time (LRDY-previous LRDP) (16) Previous lighter type
OUTPUT	BCD	Output	Contains system messages, if any.

MAIN PREV

SUBROUTINE CONVERT (1)

Purpose

Converts day, shift, hours and minutes to minutes from the beginning of the test.

Calling Sequence

CALL CONVERT (NDAY, NSHIFT, NHM, MINS)

<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
NDAY	Integer	-	Input	Julian Date.
NSHIFT	Integer	-	Input	Shift 1 or shift 2.
NHM	Integer	-	Input	Time of day in hours and minutes (military time).
MINS	Integer	-	Output	Minutes elapsed from the beginning of the test.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
None	None	None

Subroutines Called

None

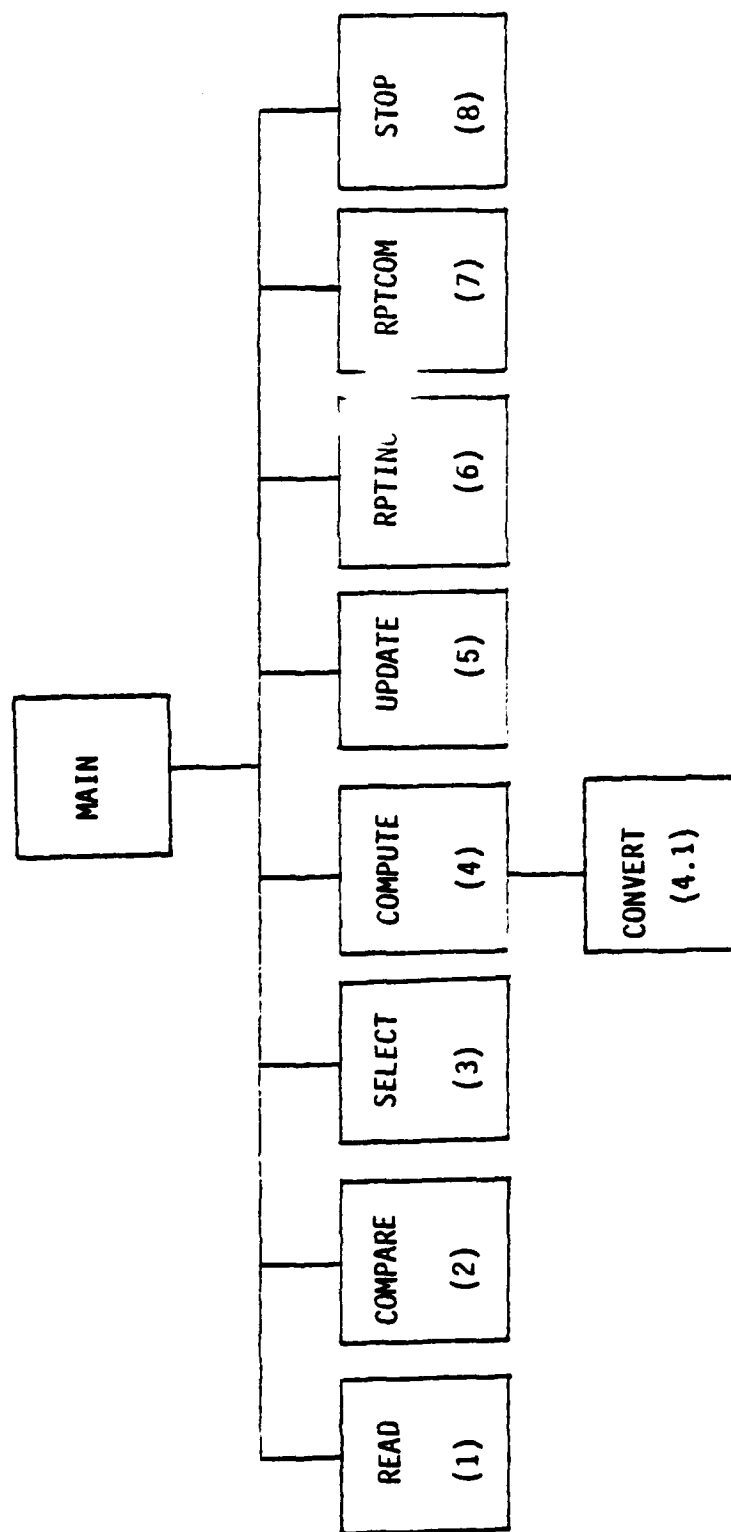


FIGURE 3.10. COMBINE STRUCTURE

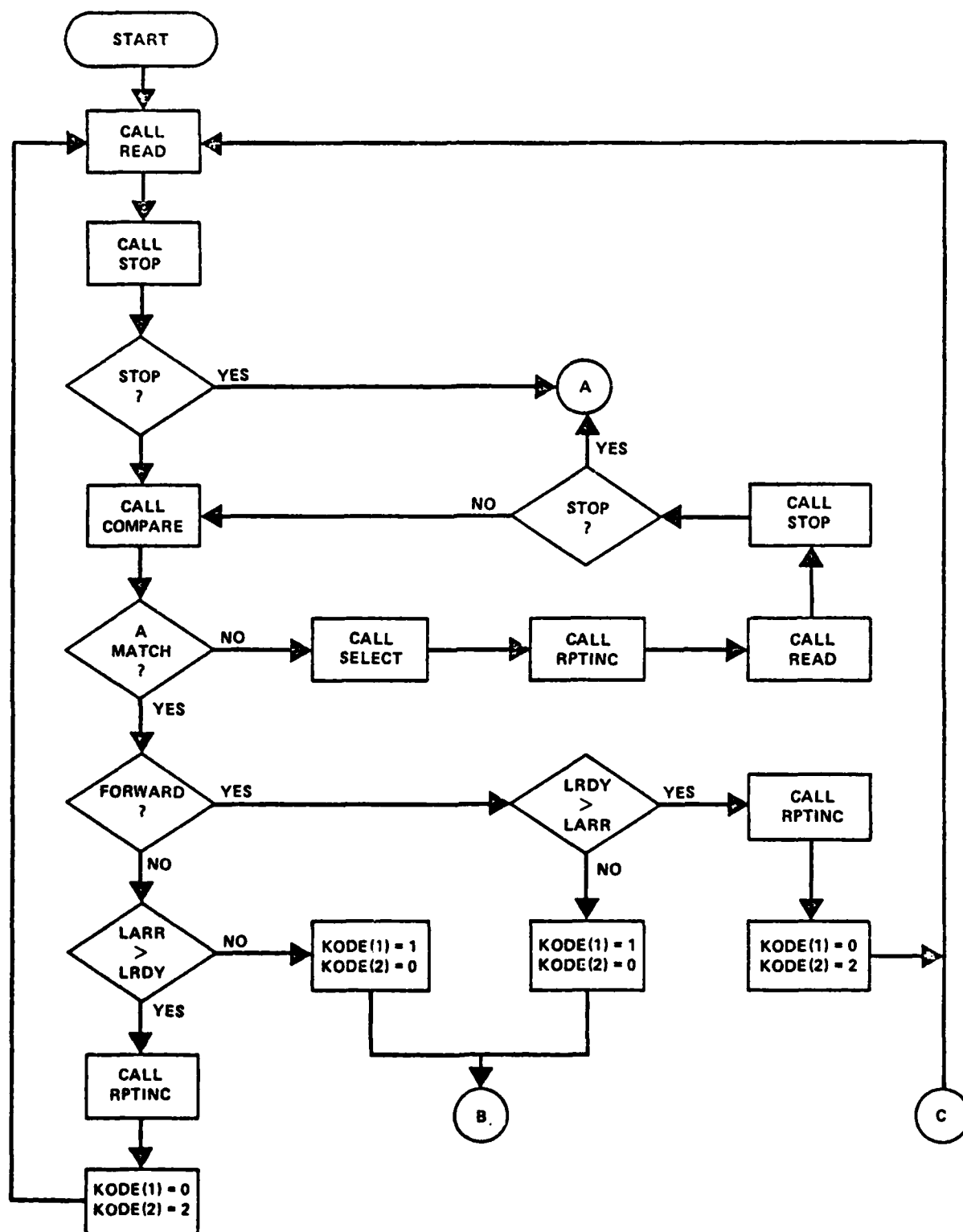


FIGURE 3.11. MAIN PROGRAM (SHRGEN) FLOW DIAGRAM

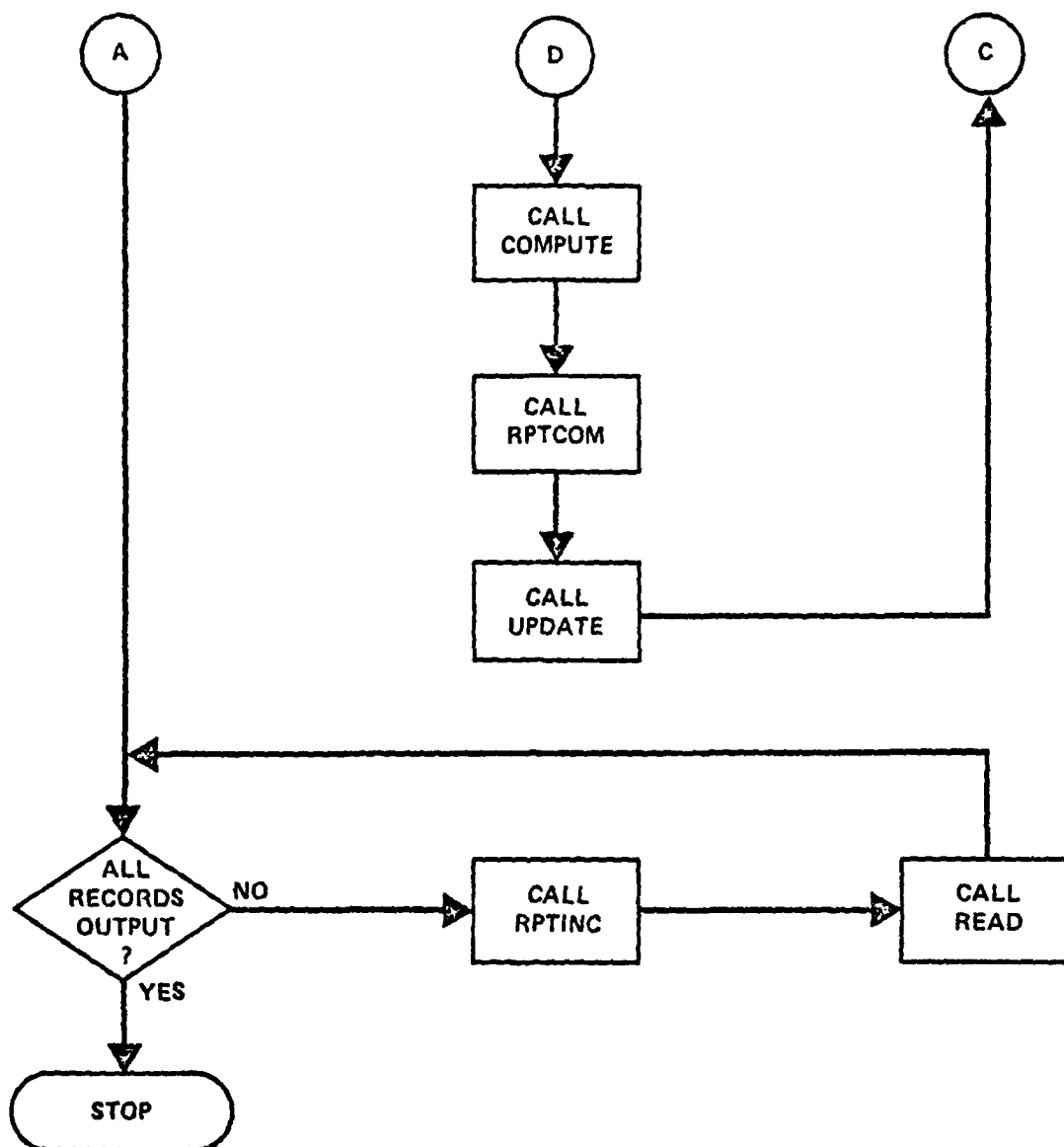


FIGURE 3.13. (CON'T) MAIN PROGRAM (SHRGEN) FLOW DIAGRAM

MAIN PROGRAM COMBINE

Purpose

To combine a lighter at the ship record with a lighter at the shore record.

Program Statement

PROGRAM COMBINE (SHIP, SHORE, OUTI3, OUIC3, OUTBN3, OUTPUT)

<u>FILE NAME</u>	<u>TYPE</u>	<u>USE</u>	<u>DESCRIPTION</u>
SHIP	Binary	Input	A SHIP record contains:
			(1) Facility ID
			(2) Julian Date
			(3) Shift
			(4) Lighter type
			(5) Lighter ID
			(6) Lighter cycle
			(7) Lighter position
			(8) Forward or retrograde (F/R)
			(9) Lighter event time (LRDY)
			(10) Lighter event time (LRDP)
			(11) Lighter at ship time (LRDP-LRDY)
SHORE	Binary	Input	A SHORE record contains:
			(1) Facility ID
			(2) Julian Date
			(3) Shift
			(4) Lighter type
			(5) Lighter ID
			(6) Lighter cycle
			(7) Lighter event time (LARR)
			(8) Lighter event time (LPSN)
			(9) Lighter event time (LRDY)
			(10) Lighter event time (LRDP)
			(11) Lighter in position time (LPSN-LARR)
			(12) Lighter ready time (LRDY-LPSN)
			(13) Lighter at crane time (LRDP-LRDY)
			(14) Lighter succession time (LRDY-previous LRDP)
			(15) Previous Lighter type

<u>FILE NAME</u>	<u>TYPE</u>	<u>USE</u>	<u>DESCRIPTION</u>
OUTI3	BCD	Output	Any SHIP or SHORE record that doesn't have a matching record in the other file.
OUTC3	BCD	Output	(1) Ship facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Forward or retrograde (F/F) (9) Lighter ship event time (LRDY) (10) Lighter ship event time (LRDP) (11) Lighter at ship time (LRDP-LRDY) (12) Shore facility ID (13) Lighter shore event time (LARR) (14) Lighter under way time (15) Lighter shore event time (LPSN) (16) Lighter in position time (LPSN-LARR) (17) Lighter shore event time (LRDY) (18) Lighter ready time (LRDY-LSPN) (19) Lighter shore event time (LRDP) (20) Lighter at crane time (LRDP-LRDY) (21) Lighter succession time (22) Previous lighter type
OUTBN3	Binary	Output	Contains the same record as OUTC3 plus minutes elapsed from the beginning of the test.
OUTPUT	BCD	Output	Contains systems messages, if any.

MAIN COMBINE

SUBROUTINE READ (1)

Purpose

To read a record from the ship and/or shore file.

Calling Sequence

CALL READ (KODE, NSHIP, NSHORE, NTIME1, NTIME2, NTIME3, NEND)

<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
KODE	Integer	2	Input	(1) Read a ship record. (2) Read a shore record.
NSHIP	Integer	7	Output	Gives lighter at the ship parameters: (1) Facility ID (2) Julian data (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position
NSHORE	Integer	6	Output	Gives lighter at the shore parameters: (1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle
NTIME1	Integer	4	Output	Gives lighter at the ship parameters (1) Forward or retrograde (2) Lighter event time (LRDY) (3) Lighter event time (LRDP) (4) Lighter at ship time (LRDP-LRDY)
NTIME2	Integer	7	Output	Gives lighter at the shore parameters: (1) Lighter event time (LARR) (2) Lighter event time (LPSN) (3) Lighter event time (LRDY) (4) Lighter event time (LRDP) (5) Lighter in position time (LPSN-LARR) (6) Lighter ready time (LRDY-LPSN) (7) Lighter at crane time (LRDP-LRDY)
NTIME3	Integer	3	Output	(1) Lighter underway time (2) Lighter success on time (3) Previous lighter type
NEND	Integer	2	Output	Equals EOF when end-of-file is reached.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
TBLUNIT	NUNIT1, NUNIT2, NUNIT3, NUNIT4, NUNIT5, NUNIT6	None

Subroutines Called

None

MAIN COMBINE

SUBROUTINE COMPARE (2)

Purpose

To determine if two records match on two match keys (lighter type and lighter identification).

Calling Sequence

CALL COMPARE (N1, N2, N3, N4, MATCH, NFAIL1, NFAIL 2)

<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
N1	Integer	-	Input	Lighter type at the ship.
N2	Integer	-	Input	Lighter ID at the ship.
N3	Integer	-	Input	Lighter type at the shore.
N4	Integer	-	Input	Lighter ID at the shore.
MATCH	Logical	-	Output	Match equals true if a match was found. Match equals false if the match failed.
NFAIL1	Integer	-	Output	Contains the NSHIP match key where the match failed.
NFAIL2	Integer	-	Output	Contains the NSHORE match key where the match failed.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
None	None	None

Subroutines Called

None

MAIN COMBINE

SUBROUTINE SELECT (3)

Purpose

To determine which record to read.

Calling Sequence

CALL SELECT (NFAIL1, NFAIL2, KODE)

<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
NFAIL1	Integer	-	Input	Contains the NSHIP match key where the match failed.
NFAIL2	Integer	-	Input	Contains the NSHORE match key where the match failed.
KODE	Integer	2	Output	(1) Read a ship record. (2) Read a shore record.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
None	None	None

Subroutines Called

None

MAIN COMBINE

SUBROUTINE COMPUTE (4)

Purpose

To compute lighter succession time at shore; to compute lighter underway time.

Calling Sequence

CALL COMPUTE (NSHIP, NSHORE, NTIME1, NTIME2, NFWD, NTIME3)

<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
NSHIP	Integer	7	Input	Gives lighter at the ship parameters: (1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position
NSHORE	Integer	6	Input	Gives lighter at shore the parameters: (1) Facility ID (2) Julian date (3) Shift (4) Lighter time (5) Lighter ID (6) Lighter cycle
NTIME1	Integer	4	Input	Gives lighter at the ship parameters: (1) Forward or retrograde (2) Lighter event time (LRDY) (3) Lighter event time (LRDP) (4) Lighter at ship time (LRDP-LRDY)
NTIME2	Integer	7	Input	Gives lighter at the shore parameters: (1) Lighter event time (LARR) (2) Lighter event time (LPSN) (3) Lighter event time (LRDY) (4) Lighter event time (LRDP) (5) Lighter in position time (LPSN-LARR) (6) Lighter ready time (LRDY-LPSN) (7) Lighter at crane time (LRDP-LRDY)
NTIME3	Integer	3	Output	(1) Lighter underway time (2) Lighter succession time (3) Previous lighter type

Common Blocks

Block Name

Input

Output

None

None

None

Subroutines Called

CONVERT

MAIN COMBINE

SUBROUTINE CONVERT (4.1)

Purpose

To convert days, shifts, hours and minutes into minutes from the beginning of the test.

Calling Sequence

CALL CONVERT (NDAY, NSHIFT, NHM, MINS)

<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
NDAY	Integer	-	Input	Julian date
NSHIFT	Integer	-	Input	Shift
NHM	Integer	-	Input	Time of day in hours and minutes (Military time).
MINS	Integer	-	Output	Minutes elapsed from the beginning of the test.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
None	None	None

Subroutines Called

None

MAIN COMBINE

SUBROUTINE UPDATE (5)

Purpose

To save either a ship record or a shore record, depending on the value of KODE.

Calling Sequence

CALL UPDATE (KODE, NSHIP, LSHIP, NSHORE, LSHORE, NTIME1, LTIME1, NTIME2, LTIME2, NTIME3, LTIME3)

<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
KODE	Integer	2	Input	(1) KODE(1) = 1 save a ship record (2) KODE(2) = 2 save a shore record
NSHIP	Integer	7	Input	Gives lighter at the ship parameters: (1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position
LSHIP	Integer	7	Output	Same as NSHIP
NSHORE	Integer	6	Input	Gives lighter at shore the parameters: (1) Facility ID (2) Julian date (3) Shift (4) Lighter time (5) Lighter ID (6) Lighter cycle
LSHORE	Integer	6	Output	Same as NSHORE
NTIME1	Integer	4	Output	Gives lighter at the ship parameters: (1) Forward or retrograde (2) Lighter event time (LRDY) (3) Lighter event time (LRDP) (4) Lighter at ship time (LRDP-LRDY)
LTIME1	Integer	4	Output	Same as LTIME
NTIME2	Integer	7	Output	Gives lighter at the shore parameters: (1) Lighter event time (LARR) (2) Lighter event time (LPSN) (3) Lighter event time (LRDY) (4) Lighter event time (LRDP) (5) Lighter in position time (LPSN-LARR) (6) Lighter ready time (LRDY-LPSN) (7) Lighter at crane time (LRDP-LRDY)

LTIME2	Integer	7	Output	
NTIME3	Integer	3	Input	(1) Lighter underway time (2) Lighter succession time (3) Previous lighter type
LTIME3	Integer	3	Output	

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
None	None	None

Subroutines Called

None

MAIN COMBINE

SUBROUTINE RPTINC (6)

Purpose

To write a non-matching record.

Calling Sequence

CALL RPTINC (KODE, NSHIP, NSHORE, NTIME1, NTIME2)

<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
KODE	Integer	2	Input	(1) Write a ship record (2) Write a non-matching shore record
NSHIP	Integer	7	Input	Gives lighter at the ship parameters: (1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position
NSHORE	Integer	6	Input	Gives lighter at shore the parameters: (1) Facility ID (2) Julian date (3) Shift (4) Lighter time (5) Lighter ID (6) Lighter cycle
NTIME1	Integer	4	Input	Gives lighter at the ship parameters: (1) Forward or retrograde (2) Lighter event time (LRDY) (3) Lighter event time (LRDP) (4) Lighter at ship time (LRDP-LRDY)
NTIME2	Integer	7	Input	Gives lighter at the shore parameters: (1) Lighter event time (LARR) (2) Lighter event time (LPSN) (3) Lighter event time (LRDY) (4) Lighter event time (LRDP) (5) Lighter in position time (LPSN-LARR) (6) Lighter ready time (LRDY-LPSN) (7) Lighter at crane time (LRDP-LRDY)

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
TBLUNIT	NUNIT1, NUNIT2, NUNIT3, NUNIT4, NUNIT5, NUNIT6	None

Subroutines Called

None

MAIN COMBINE

SUBROUTINE RPTCOM (7)

Purpose

To write a combined record both in binary and BCD.

Calling Sequence

CALL RPTCOM (NSHIP, NSHORE, NTIME1, NTIME2, NTIME3, NSORT)

<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
NSHIP	Integer	7	Input	Gives lighter at the ship parameters: (1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position
NSHORE	Integer	6	Input	Gives lighter at shore the parameters: (1) Facility ID (2) Julian date (3) Shift (4) Lighter time (5) Lighter ID (6) Lighter cycle
NTIME1	Integer	4	Input	Gives lighter at the ship parameters: (1) Forward or retrograde (2) Lighter event time (LRDY) (3) Lighter event time (LRDP) (4) Lighter at ship time (LRDP-LRDY)
NTIME2	Integer	7	Input	Gives lighter at the shore parameters: (1) Lighter event time (LARR) (2) Lighter event time (LPSN) (3) Lighter event time (LRDY) (4) Lighter event time (LRDP) (5) Lighter in position time (LPSN-LARR) (6) Lighter ready time (LRDY-LPSN) (7) Lighter at crane time (LRDP-LRDY)
NTIME3		3	Input	(1) Lighter underway time (2) Lighter succession time (3) Previous lighter type
NSORT			Input	Time from the beginning of the test based on LRDY at the ship.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
TBLUNIT	NUNIT1, NUNIT2, NUNIT3, NUNIT4, NUNIT5, NUNIT6	None

Subroutines Called

None

MAIN COMBINE

SUBROUTINE STOP (8)

Purpose

To determine if EOF is in any file or in all files. Sets KODE to write an incomplete record.

Calling Sequence

CALL STOP (NEND, NSTOP, NSTOP1, KODE)

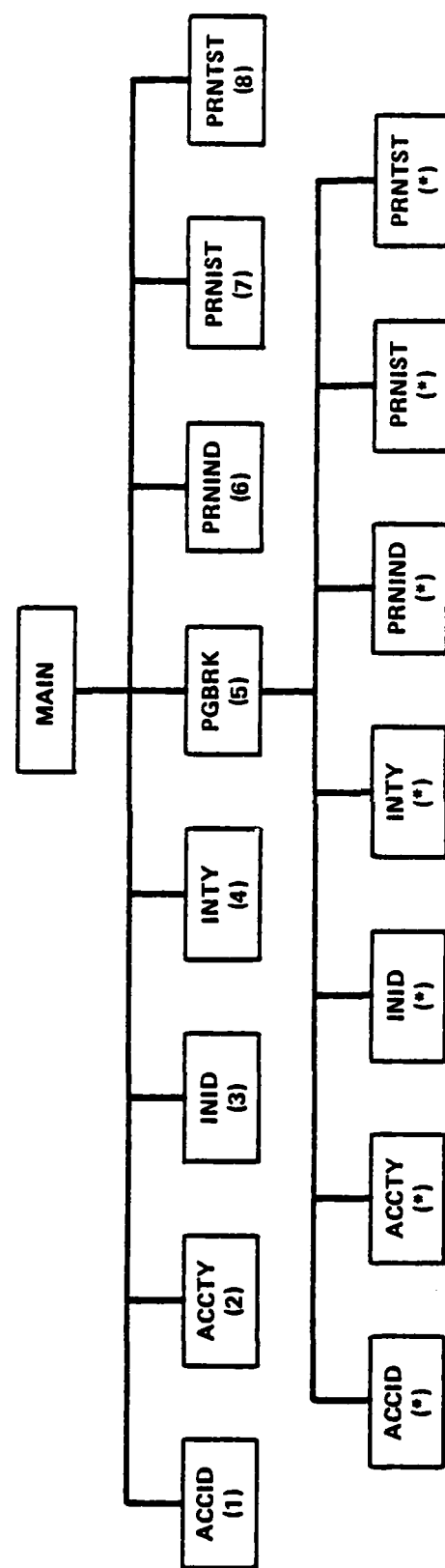
<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
NEND	Integer	2	Input	Equals EOF when end of file is reached.
NSTOP	Logical	-	Output	When NSTOP is false, all files contain data. When NSTOP is true, at least one file is out of data.
NSTOP1	Logical	-	Output	When NSTOP1 is true, all files are out of data.
KODE	Integer	2	Output	(1) Read a ship file. (2) Read a shore file.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
None	None	None

Subroutines Called

None



* THIS SUBROUTINE IS ALSO CALLED BY MAIN.

FIGURE 3.14. WTRPT STRUCTURE

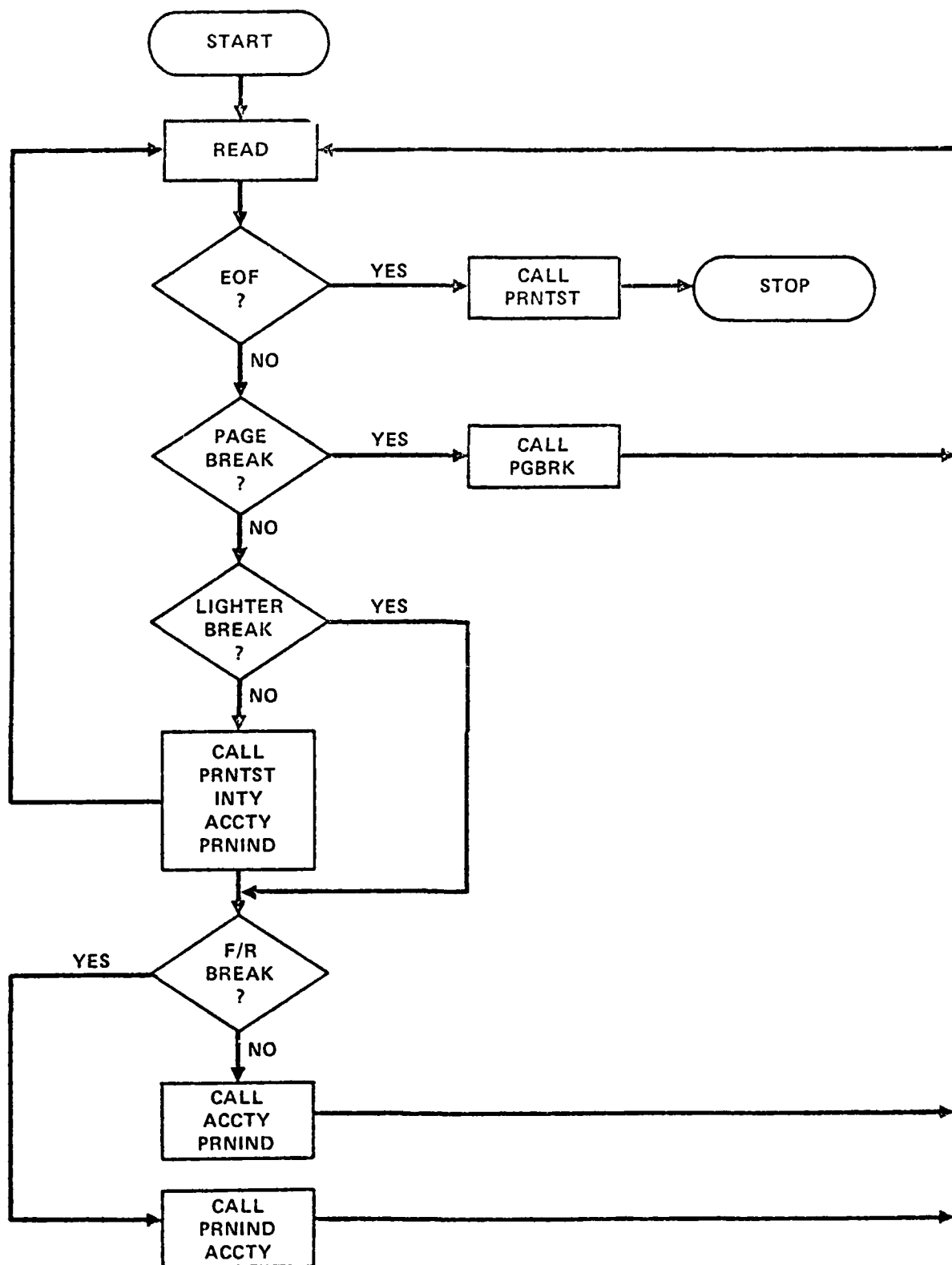


FIGURE 3.15. MAIN PROGRAM (WRTRPT) FLOW DIAGRAM

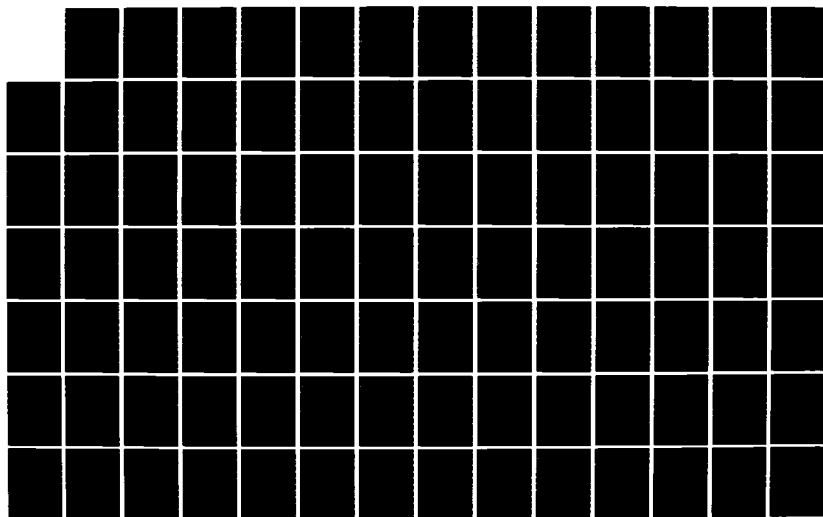
AD-A131 715

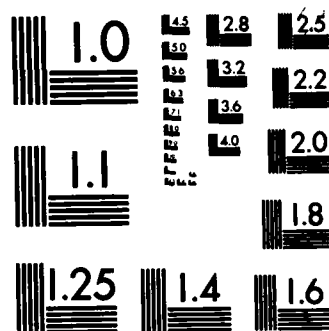
JOINT LOGISTICS-OVER-THE-SHORE (LOTS) MAIN TEST
AUTOMATED DATA BASE REDUCTION PROGRAMS(U) ORI INC
SILVER SPRING MD H CASEY ET AL. MAR 79 ORI-TR-1477
MDA903-75-C-0016 F/G 9/2

2/

UNCLASSIFIED

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

MAIN PROGRAM WTRPT

Purpose

To read the sorted ship to shore times file, compute statistics and write report. For each ship facility, date and shift, statistics are accumulated on individual lighters and lighter types.

Program Statement

PROGRAM WTRPT (TAPE1, OUTPUT)

<u>FILE NAME</u>	<u>TYPE</u>	<u>USE</u>	<u>DESCRIPTION</u>
TAPE1	Binary	Input	(1) Ship Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Forward or retrograde (F/R) (9) Lighter ship event time (LRDY) (10) Lighter ship event time (LRDP) (11) Lighter at ship time (LRDP-LRDY) (12) Shore Facility ID (13) Lighter shore event time (LARR) (14) Lighter under way time (15) Lighter shore event time (LPSN) (16) Lighter in position time (LPSN-LARR) (17) Lighter shore event time (LRDY) (18) Lighter ready time (LRDY-LPSN) (19) Lighter shore event time (LRDP) (20) Lighter at crane time (LRDP-LRDY) (21) Lighter succession time (LRDY - previous LRDP) (22) Previous Lighter type

MAIN WTRPT

SUBROUTINE ACCID (1)

Purpose

To accumulate individual lighter statistics

Calling Sequence

CALL ACCID (SUMID, SUMSQID, NOID, NSHIP, NSHORE)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
SUMID	Real	6	Input or Output	(1) Sum of lighter at ship times (Σ LRDP-LRDY) (2) Sum of lighter under way times (3) Sum of lighter in position times (Σ LPSN-LARR) (4) Sum of lighter ready times (Σ LRDY-LPSN) (5) Sum of lighter at crane times (Σ LRDP-LRDY) (6) Sum of lighter succession times
SUMSQID	Real	6	Input or Output	(1) - (6) Sums of squares of the same 6 items as above
NOID	Integer	3	Input or Output	(1) Number of lighter under way, lighter in position, lighter ready, and lighter at crane times included in sums (2) Number of lighter at ship times included in sums (3) Number of lighter succession times included in sums
NSHIP	Integer	11	Input	(1) Ship Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Forward or retrograde (F/R) (9) Lighter ship event time (LRDY) (10) Lighter ship event time (LRDP) (11) Lighter at ship time (LRDP-LRDY)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
NSHORE	Integer	11	Input	(1) Shore Facility ID (2) Lighter shore event time (LARK) (3) Lighter under way time (4) Lighter shore event time (LPSN) (5) Lighter in position time (LPSN-LARR) (6) Lighter shore event time (LRDY) (7) Lighter ready time (LRDY-LPSN) (8) Lighter shore event time (LRDP) (9) Lighter at crane time (LRDP-LRDY) (10) Lighter succession time (LRDY - previous LRDP) (11) Previous lighter type

Commons Blocks

None

Subroutines Called

None

MAIN WRTRPT

SUBROUTINE ACCTY (2)

Purpose

To accumulate lighter type statistics

Calling Sequence

CALL ACCTY (SUMTY, SUMSQTY, NOTYF, NOTYR, NSHIP, NSHORE)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
SUMTY	Real	6,2	Input or Output	(1,1) Sum of forward lighter at ship times (2,1) Sum of forward lighter under way times (3,1) Sum of forward lighter in position times (4,1) Sum of forward lighter ready times (5,1) Sum of forward lighter at crane times (6,1) Sum of forward lighter succession times (1,2) - (6,2) Sums of same 6 times as above except retrograde instead of forward
SUMSQTY	Real	6,2	Input or Output	Corresponding sums of squares of times above
NOTYF	Integer	3	Input or Output	(1) Number of forward lighter under way, lighter in posi- tion, lighter ready and lighter at crane times included in "forward" sums. (2) Number of forward lighter at ship times included in "forward" sums. (3) Number of forward lighter succession times included in "forward" sums.
NOTYR	Integer	3	Input or Output	(1) Number of retrograde lighter under way, lighter in posi- tion, lighter ready and lighter at crane times included in "retrograde" sums. (2) Number of retrograde lighter at ship times included in "retrograde" sums.

MAIN WTRPT

SUBROUTINE ACCTY (2) (CONTINUED)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
				(3) Number of retrograde lighter succession times included in "retrograde" sums.
NSHIP	Integer	11	Input	(1) Ship Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Forward or retrograde (F/R) (9) Lighter ship event time (LRDY) (10) Lighter ship event time (LRDP) (11) Lighter at ship time (LRDP-LRDY)
NSHORE	Integer	11	Input	(1) Shore Facility ID (2) Lighter shore event time (LARR) (3) Lighter under way time (4) Lighter shore event time (LPSN) (5) Lighter in position time (LPSN-LARR) (6) Lighter shore event time (LRDY) (7) Lighter ready time (LRDY-LPSN) (8) Lighter shore event time (LRDP) (9) Lighter at crane time (LRDP-LRDY) (10) Lighter succession time (LRDY - previous LRDP) (11) Previous lighter type

Common Blocks

None

Subroutines Called

None

MAIN WRTRPT

SUBROUTINE INID (3)

Purpose

To initialize individual lighter statistics arrays

Calling Sequence

CALL INID (SUMID, SUMSQID, NOID)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
SUMID	Real	6	Output	(1) Sum of lighter at ship times (Σ LRDP-LRDY) (2) Sum of lighter under way times (3) Sum of lighter in position times (Σ LPSN-LARR) (4) Sum of lighter ready times (Σ LRDY-LPSN) (5) Sum of lighter at crane times (Σ LRDP-LRDY) (6) Sum of lighter succession times
SUMSQID	Real	6	Output	(1) - (6) Sums of squares of the same 6 items as above
NOID	Integer	3	Output	(1) Number of lighter under way, lighter in position, lighter ready, and lighter at crane times included in sums (2) Number of lighter at ship times included in sums (3) Number of lighter succession times included in sums

Common Blocks

None

Subroutines Called

None

MAIN WRTRPT

SUBROUTINE INTY (4)

Purpose

To initialize lighter type statistics arrays

Calling Sequence

CALL INTY (SUMTY, SUMSQTY, NOTYF, NOTYR)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
SUMTY	Real	6,2	Output	(1,1) Sum of forward lighter at ship times (2,1) Sum of forward lighter under way times (3,1) Sum of forward lighter in position times (4,1) Sum of forward lighter ready times (5,1) Sum of forward lighter at crane times (6,1) Sum of forward lighter succession times (1,2) - (6,2) Sums of same 6 times as above except retrograde instead of forward
SUMSQTY	Real	6,2	Output	Corresponding sums of squares of times above
NOTYF	Integer	6,2	Output	(1) number of forward lighter under way, lighter in position, lighter ready and lighter at crane times included in "forward" sums. (2) Number of forward lighter at ship times included in "forward" sums. (3) Number of forward lighter succession times included in "forward" sums.
NOTYR	Integer	3	Output	(1) Number of retrograde lighter under way, lighter in position, lighter ready and lighter at crane times included in "retrograde" sums. (2) Number of retrograde lighter at ship times included in "retrograde" sums.

MAIN WRTRPT

SUBROUTINE INTY (4) (CONTINUED)

Common Blocks

None

Subroutines Called

None

MAIN WTRPT

SUBROUTINE PBRK (5)

Purpose

To write a new page heading when there is a "break" on facility, date or shift. In addition, calls PRNTSY to print lighter type statistics just prior to the heading and calls PRNIND to print the first individual lighter record after the heading.

Calling Sequence

CALL PGBRK (FIRST, OFAC, ODATE, OSHIFT, NSHIP, NSHORE, KPGBR, OLTRTY, OLTRID, OFR, SUMID, SUMSQID, SUMTY, SUMSQTY, NOID, NOTYF, NOTYR)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
FIRST	Logical	-	Input	First call indicator
OFAC	Real	-	Output	Current Facility ID
ODATE	Real	-	Output	Current date
OSHIFT	Real	-	Output	Current shift
NSHIP	Integer	11	Input	(1) Ship Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Forward or retrograde (F/R) (9) Lighter ship event time (LRDY) (10) Lighter ship event time (LRDP) (11) Lighter at ship time (LRDP-LRDY)
NSHORE	Integer	11	Input	(1) Shore Facility ID (2) Lighter shore event time (LARR) (3) Lighter under way time (4) Lighter shore event time (LPSN) (5) Lighter in position time (LPSN-LARR) (6) Lighter shore event time (LARDY) (7) Lighter ready time (LRDY-LPSN) (8) Lighter shore event time (LRDP) (9) Lighter at crane time (LRDP-LRDY) (10) Lighter succession time (LRDY- previous LRDP) (11) Previous lighter type

MAIN WRTRPT

SUBROUTINE PGBRK (5) (CONTINUED)

VARIABLE	TYPE	DIMENSION	USE	DESCRIPTION
KPGBR	Logical	-	Output	Page break indicator
OLTRTY	Real	-	Output	Current lighter type
OLTRID	Real	-	Output	Current lighter ID
OFR	Real	-	Output	Current forward/retrograde indicator
SUMID	Real	6	Input	(1) Sum of lighter at ship times (Σ LRDP-LRDY) (2) Sum of lighter under way times (3) Sum of lighter in position times (Σ LPSN-LARR) (4) Sum of lighter ready times (Σ LRDY-LPSN) (5) Sum of lighter at crane times (Σ LRDP-LRDY) (6) Sum of lighter succession times
SUMSQID	Real	6	Input	(1) - (6) Sums of squares of the same 6 times as above.
SUMTY	Real	6,2	Input	(1,1) Sum of forward lighter at ship times (2,1) Sum of forward lighter under way times (3,1) Sum of forward lighter in position times (4,1) Sum of forward lighter ready times (5,1) Sum of forward lighter at crane times (6,1) Sum of forward lighter succession times (1,2) - (6,2) Sums of same 6 times as above except retrograde instead of forward
SUMSQTY	Real	6,2	Input	Corresponding sums of squares of times above.

MAIN WRTRPT

SUBROUTINE PBGRK (5) (CONTINUED)

VARIABLE	TYPE	DIMENSION	USE	DESCRIPTION
NOID	Integer	3	Input	(1) Number of lighter under way, lighter in position, lighter ready, and lighter at crane times included in sums. (2) Number of lighter at ship times included in sums. (3) Number of lighter succession times included in sums.
NOTYF	Integer	3	Input	(1) Number of forward lighter under way, lighter in position, lighter ready and lighter at crane times included in "forward" sums. (2) Number of forward lighter at ship times included in "forward" sums. (3) Number of forward lighter succession times included in "forward" sums.
NOTYR	Integer	3	Input	(1) Number of retrograde lighter under way, lighter in position, lighter ready and lighter at crane times included in "retrograde" sums. (2) Number of retrograde lighter at ship times included in "retrograde" sums. (3) Number of retrograde lighter succession times included in "retrograde" sums.

Common Blocks

None

Subroutines Called

PRNTST
PRNIND
INID
INTY
ACCID
ACCTY

MAIN WRTRPT

SUBROUTINE PRNIND (6)

Purpose

To write an individual lighter ship to shore record (i.e., one line in the report).

Calling Sequence

CALL PRNIND (NSHIP, NSHORE)

VARIABLE	TYPE	DIMENSION	USE	DESCRIPTION
NSHIP	Integer	11	Input	(1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Forward or retrograde (F/R) (9) Lighter ship event time (LRDY) (10) Lighter ship event time (LRDP) (11) Lighter at ship time (LRDP-LRDY)
NSHORE	Integer	11	Input	(1) Shore Facility ID (2) Lighter shore event time (LARR) (3) Lighter under way time (4) Lighter shore event time (LPSN) (5) Lighter in position time (LPSN-LARR) (6) Lighter shore event time (LRDY) (7) Lighter ready time (LRDY-LPSN) (8) Lighter shore event time (LRDP) (9) Lighter at crane time (LRDP-LRDY) (10) Lighter succession time (LRDY-previous LRDP) (11) Previous lighter type

Common Blocks

None

Subroutines Called

None

MAIN WRTRPT

SUBROUTINE PRNIST (7)

Purpose

To compute and print individual lighter statistics

Calling Sequence

CALL PRNIST (SUMID, SUMSQID, NOID)

VARIABLE	TYPE	DIMENSION	USE	DESCRIPTION
SUMID	Read	6	Input	(1) Sum of lighter at ship times (Σ LRDP-LRDY) (2) Sum of lighter under way times (3) Sum of lighter in position times (Σ LPSN-LARR) (4) Sum of lighter ready times (Σ LRDY-LPSN) (5) Sum of lighter at crane times (Σ LRDP-LRDY) (6) Sum of lighter succession times.
SUMSQID	Real	6	Input	(1) - (6) Sums of squares of the same 6 times as above.
NOID	Integer	3	Input	(1) Number of lighter under way, lighter in position, lighter ready, and lighter at crane times included in sums. (2) Number of lighter at ship times included in sums. (3) Number of lighter succession times included in sums.

Common Blocks

None

Subroutines Called

None

MAIN WRTRPT

SUBROUTINE PRNTST (8)

Purpose

To computer and print lighter type statistics

Calling Sequence

CALL PRNTST (SUMTY, SUMSQTY, NOTYF, NOTYR)

VARIABLE	TYPE	DIMENSION	USE	DESCRIPTION
SUMTY	Real	6,2	Input	(1,1) Sum of forward lighter at ship times (2,1) Sum of forward lighter under way times (3,1) Sum of forward lighter in position times (4,1) Sum of forward lighter ready times (5,1) Sum of forward lighter at crane times (6,1) Sum of forward lighter succession times (1,2) - (6,2) Sums of same 6 times as above except retrograde instead of forward
SUMSQTY	Real	6,2	Input	Corresponding sums of squares of times above.
NOTYF	Integer	3	Input	(1) Number of forward lighter under way, lighter in position, lighter ready and lighter at crane times included in "forward" sums. (2) Number of forward lighter at ship times included in "forward" sums. (3) Number of forward lighter succession times included in "forward" sums.
NOTYR	Integer	3	Input	(1) Number of retrograde lighter under way, lighter in position, lighter ready and lighter at crane times included in "retrograde" sums. (2) Number of retrograde lighter at ship times included in "retrograde" sums. (3) Number of retrograde lighter succession times included in "retrograde" sums.

MAIN WRTRPT

SUBROUTINE PRNTST (8) (CONTINUED)

Common Blocks

None

Subroutines Called

None

IV. LIGHTER AND CARGO CYCLE TIMES PROGRAM

INTRODUCTION

The Lighter and Cargo Cycle Times Program gives times relating to the interaction of lighters and their cargo at the ship facilities. For each lighter it gives:

- 1) Lighter type
- 2) Lighter ID
- 3) Lighter cycle number
- 4) Lighter position at the ship
- 5) Direction of cargo movement (forward or retrograde)
- 6) Lighter ready at the ship facility (LRDY)
- 7) Lighter ready to deport the ship
- 8) Lighter at the ship line (LRDP-LRDY)
- 9) Lighter succession time (LRDY-previous LRDP)
- 10) Lighter cycle time (LRDP-previous LRDP)

Data for the cargo on each lighter includes:

- 1) Cargo type
- 2) Cargo ID
- 3) Crane operator ID
- 4) Lifting device over container time (LDOC)
- 5) LDOC - previous LDUN (see 15 below for description of LDUN)
- 6) Lifting device lock time (LDLK)
- 7) LDLK-LDOC
- 8) Container lifted time (LIFT)
- 9) LIFT-LDLK
- 10) Container begin position time (CPSN)
- 11) CPSN-LIFT
- 12) Container land time (LAND)

- 13) LAND-CPSN
- 14) Lifting device unlock time (LDUN)
- 15) LDUN-LAND
- 16) Delay start, if any
- 17) Delay type, if any

The number of samples, mean, and standard deviation are calculated and output for all time differences. The output data is arranged by facility ID, date, shift, lighter type, lighter ID, direction of cargo movement, and time.

A sample output is given in Figure 4.1.

PROGRAM DESCRIPTION

The Lighter and Cargo Cycles program consists of three System 2000 retrieval programs, seven main programs and 37 subprograms, six sorts and two merges. The main programs, sorts and merges are combined into one program by control statements. A flow chart of the individual steps is given in Figure 4.2.

A description of each program is given starting with the three System 2000 retrieval programs and followed by the seven main programs. The description of each main program contains a structure diagram, a flow chart of the main program followed by a write-up of each main program and its subroutines.

LIGHTER/CARGO CYCLES TIMES									
FACILITY IN YCDE DATE AUG 15 SP1ST 1									
LTR	LTB	C	T	LADP	LRDY	LRDN	LRDP	LRDP	LRDP
TYPE	ID	VS	R	LRDY	LRDY	LRDY	LRDY	LRDY	LRDY
LACV 3001	1	CF	R	1245	10	1	11	MILV 3707	12 1341
LACV 3001	2	CF	R	1514	15	14	30	MILV 3749	12 1516
LACV 3002	1	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	2	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	3	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	4	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	5	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	6	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	7	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	8	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	9	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	10	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	11	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	12	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	13	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	14	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	15	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	16	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	17	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	18	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	19	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	20	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	21	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	22	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	23	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	24	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	25	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	26	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	27	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	28	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	29	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	30	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	31	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	32	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	33	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	34	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	35	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	36	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	37	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	38	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	39	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	40	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	41	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	42	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	43	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	44	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	45	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	46	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	47	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	48	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	49	CF	R	1430	12	9	21	MILV 3650	12 1528
LACV 3002	50	CF	R	1430	12	9	21	MILV 3650	12 1528

FIGURE 4.1. LIGHTER AND CARGO CYCLE TIMES PROGRAM SAMPLE OUTPUT

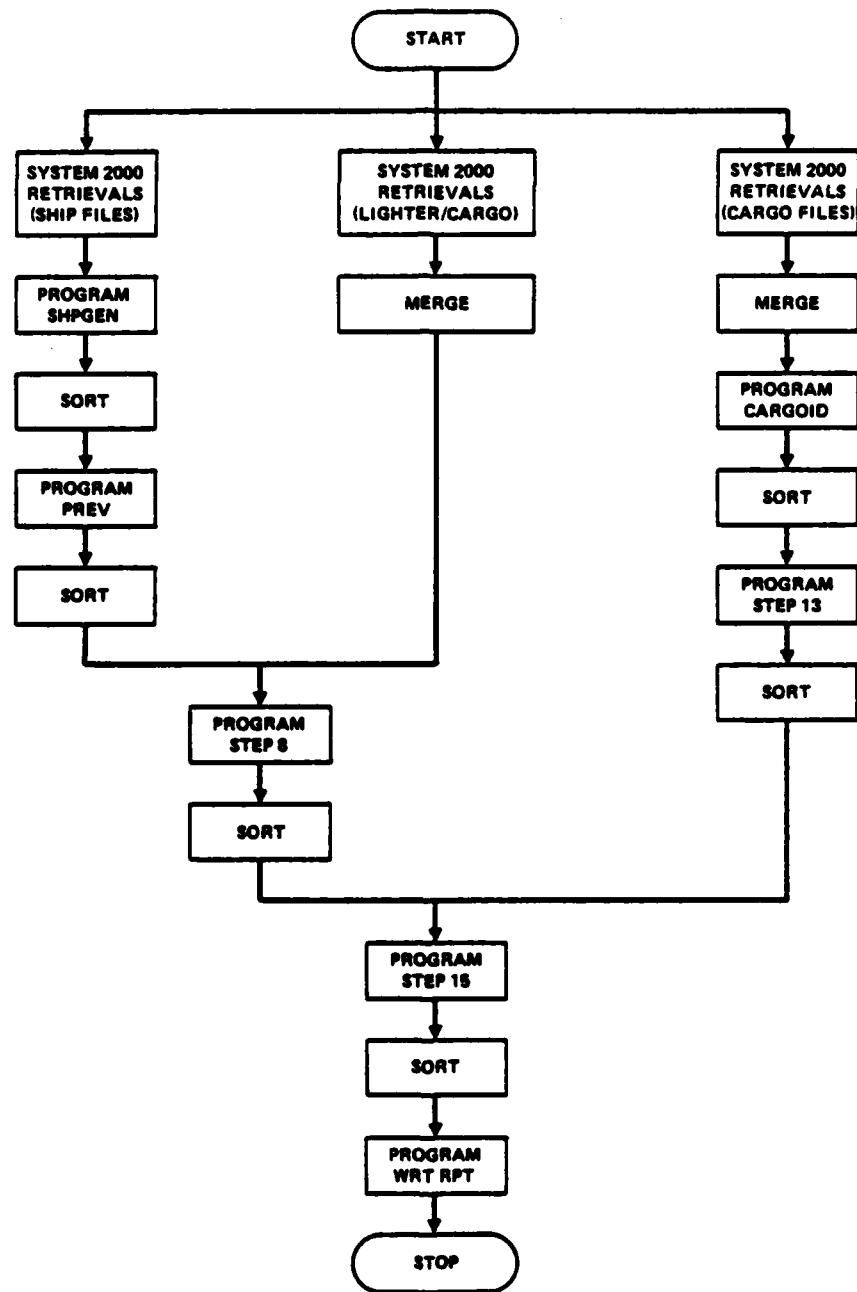


FIGURE 4.2. FLOW CHART OF THE STEPS IN THE LIGHTER AND CARGO CYCLE TIMES PROGRAM

SYSTEM 2000 RETRIEVALS FOR THE LIGHTER/CARGO FILES

Purpose

To retrieve lighter/cargo data from the LOTS system 2000 data base and to store the data on two files.

<u>FILE NAME</u>	<u>TYPE</u>	<u>USE</u>	<u>DESCRIPTION</u>
UNIT5	BCD	Output	(1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Cargo received type (9) Cargo received ID
UNIT6	BCD	Output	(1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Cargo discharged type (9) Cargo discharged ID

SYSTEM 2000 RETRIEVALS FOR THE CARGO TIMES FILE

Purpose

To retrieve cargo times data from the LOTS System 2000 data base and to store the data on two files.

<u>FILE NAME</u>	<u>TYPE</u>	<u>USE</u>	<u>DESCRIPTION</u>
UNIT9A	BCD	Output	(1) Facility ID (2) Julian date (3) Shift (4) Crane operator (5) Receiving carrier type (6) Receiving carrier ID (7) Cargo ID (8) Cargo cycle (9) Cargo event name (10) Cargo event time (11) Delay type (12) Delay category (13) Delay duration
UNIT9B	BCD	Output	(1) Facility ID (2) Julian date (3) Shift (4) Crane operator (5) Discharging carrier type (6) Discharging carrier type (7) Cargo ID (8) Cargo cycle (9) Cargo event name (10) Cargo event time (11) Delay type (12) Delay category (13) Delay duration

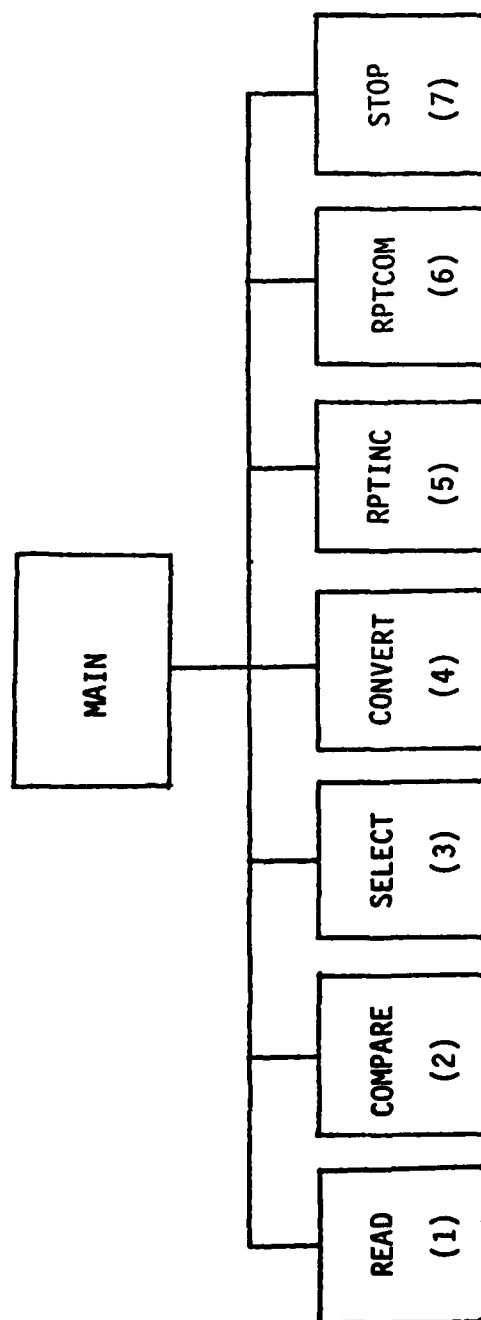


FIGURE 4.3. SHPGEN STRUCTURE

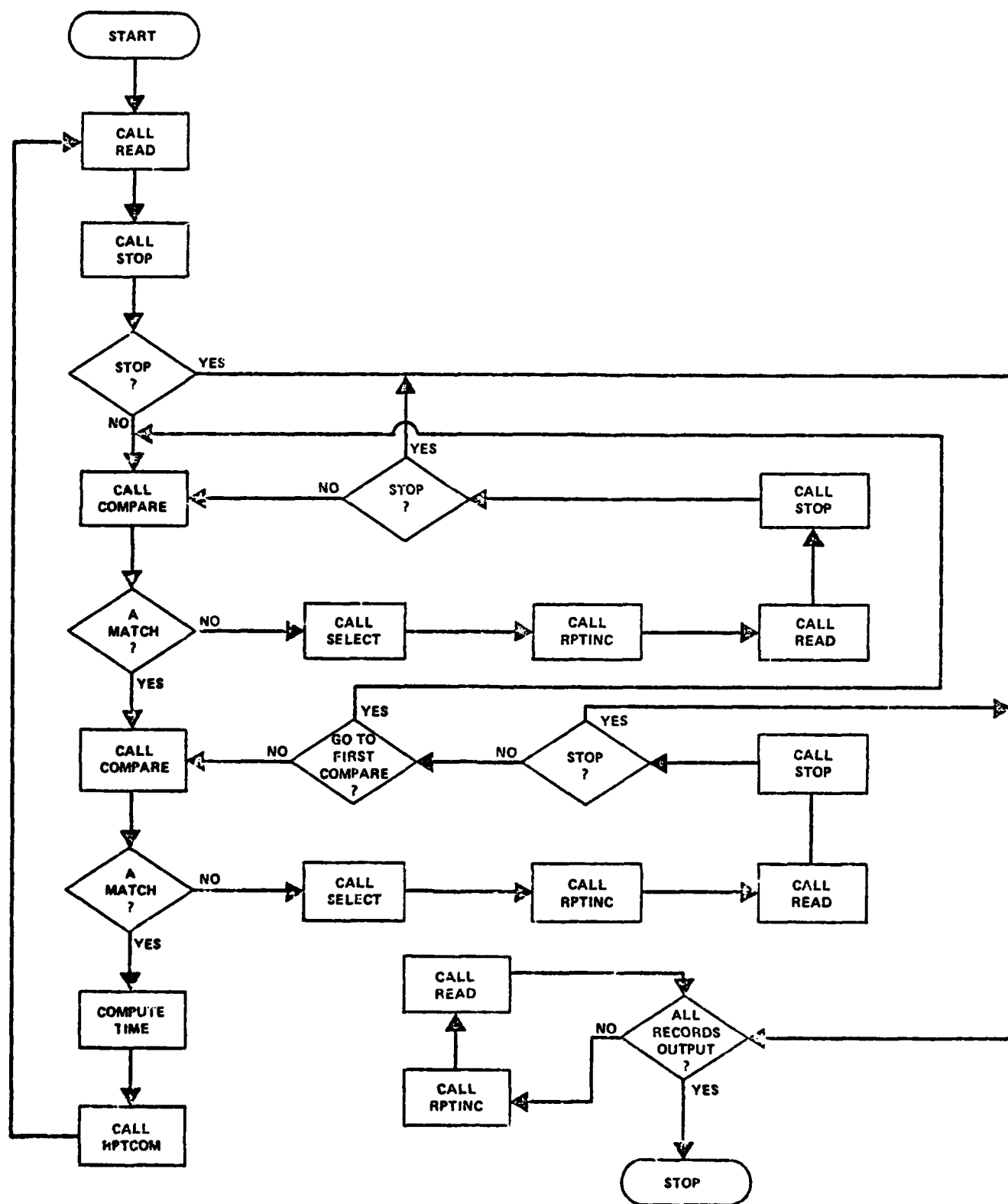


FIGURE 4.4. MAIN PROGRAM (SHPGEN) FLOW DIAGRAM

MAIN PROGRAM SHPGEN

Purpose

To combine three records, contained in the System 2000 ship files, into one record.

Program Statement

PROGRAM SHPGEN (LRDY1, LRDP1, FR1, OUTI1, OUTC1, OUTPUT, OUTBN1)

<u>FILE NAME</u>	<u>TYPE</u>	<u>USE</u>	<u>DESCRIPTION</u>
LRDY1	BCD	Input	A LRDY1 record contains: (1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Lighter event time (LRDY)
LRDP1	BCD	Input	A LRDP1 records contains: (1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter event time (LRDP)
NFR	BCD	Input	A NFR record contains: (1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Foreward or retrograde (F/R)
OUTI1	BCD	Output	Any LRDY1, LRDP2 or NFR record that doesn't have matching records in the other two files.

<u>FILE NAME</u>	<u>TYPE</u>	<u>USE</u>	<u>DESCRIPTION</u>
OUTC1	BCD	Output	A OUTC1 record contains: (1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Forward or retrograde (F/R) (9) Lighter event time (LRDY) (10) Lighter event time (LRDP) (11) Lighter Ship time (LRDP-LRDY) (12) Minutes elapsed from the beginning of the test.
OUTPUT	BCD	Output	Contains system messages, if any.
OUTBN1	Binary	Output	Contains the same record as OUTC1.

MAIN SHPGEN

SUBROUTINE READ (1)

Purpose

To read one record from one, two or three input files depending on the value of CODE.

Calling Sequence

CALL READ (CODE, NLRDY, NLRDP, NFR, NLGTPS, NPAM, NFIRST, NEND)

<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
KODE	Integer	3	Input	(1) Flag to read a file 1 record. (2) Flag to read a file 2 record. (3) Flag to read a file 3 record.
NFIRST	Logical	-	Input	Equals false on first call and true on additional calls.
NLRDY	Integer	6	Output	First six parameters of input file 1 record, i.e., (1) Facility ID (2) Date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle
NLRDP	Integer	6	Output	First six parameters of input file 2 record, i.e., (1) Facility ID (2) Date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle
NFR	Integer	6	Output	First six parameters of input file 3 record, i.e., (1) Facility ID (2) Date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle
NLGTPS	Integer	-	Output	Seventh parameter of input file 1 record; i.e., lighter position.
NPAM	Integer	3	Output	Last parameter of each of the three input file records, i.e., (1) File 1 record-lighter event time (LRDY) (2) File 2 record-lighter event time (LRDY) (3) File 3 record-forward or retrograde movement (F/R)
NEND	Integer	3	Output	Equals EOF when end-of-file is reached.

Common Blocks

<u>Block Name</u>	<u>Inputs</u>	<u>Output</u>
TBLUNIT	NUNIT1, NUNIT2, NUNIT3, NUNIT4, NUNIT5, NUNIT6, NUNIT7	None

Subroutines Called

None

MAIN SHPGEN

SUBROUTINE COMPARE (2)

Purpose

To determine if two records match on six match keys. If there is no match, it returns to the main program the first match key where the match failed.

Calling Sequence

CALL COMPARE (NTEST1, NTEST2, MATCH, NFAIL1, NFAIL2)

<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
NTEST1	Integer	6	Input	Contains the lighter ready record (six match keys).
NTEST2	Integer	6	Input	Contains the lighter ready to depart or forward/retrograde record (six match keys).
MATCH	Logical	-	Input	MATCH equals true if a match was found. MATCH equals false if the match failed.
NFAIL1	Integer	-	Output	Contains the first lighter ready match key where the match failed.
NFAIL2	Integer	-	Output	Contains the first lighter ready or the first forward/retrograde match key where the match failed.

Common Blocks

<u>Block Name</u>	<u>Inputs</u>	<u>Output</u>
None	None	None

Subroutines Called

None

MAIN SHPGEN

SUBROUTINE SELECT (3)

Purpose

To determine which record to read.

Calling Sequence

CALL SELECT (KEY, NFAIL1, NFAIL2, KODE)

<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
KEY	Integer	-	Input	When KEY equals 1, the match between a lighter ready record and a lighter ready to depart record failed. When KEY equals 2, the match between a lighter ready record and a forward/retrograde record failed.
NFAIL1	Integer	-	Input	Contains the first lighter ready match key where the match failed.
NFAIL2	Integer	-	Input	Contains the first lighter ready or the first foreward/retrograde match key where the match failed.
KODE	Integer	3	Output	If match failed: (1) Flag to indicate an incomplete file 1 record. (2) Flag to indicate an incomplete file 2 record. (3) Flag to indicate an incomplete file 3 record.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
None	None	None

Subroutines Called

None

MAIN SHPGEN

SUBROUTINE CONVERT (4)

Purpose

Converts day, shift, hours, and minutes to minutes from the beginning of the test.

Calling Sequence

CALL CONVERT (NDAY, NSHIFT, NHM, MINS)

<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
NDAY	Integer	-	Input	Julian date.
NSHIFT	Integer	-	Input	Shift 1 or shift 2.
NHM	Integer	-	Input	Time of day in hours and minutes (military time).
MINS	Integer	-	Output	Minutes elapsed from the beginning of the test.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
None	None	None

Subroutines Called

None

MAIN SHPGEN

SUBROUTINE RPTINC (5)

Purpose

To write an incomplete record.

Calling Sequence

CALL RPTINC (KODE, NLRDY, NLRDP, NFR, NLGTPS, NPAM, NFINC)

<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
KODE	Integer	3	Input	(1) Flag to write a file 1 record. (2) Flag to write a file 2 record. (3) Flag to write a file 3 record.
NLRDY	Integer	6	Input	First six parameters of input file 1 record, i.e., (1) Facility ID (2) Date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle
NLRDP	Integer	6	Input	First six parameters of input file 2 record, i.e., (1) Facility ID (2) Date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle
NFR	Integer	6	Input	First six parameters of input file 3 record, i.e., (1) Facility ID (2) Date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle
NLGTPS	Integer	-	Input	Seventh parameter of input file 1 record, i.e., lighter position.
NPAM	Integer	3	Input	Last parameter of each of the three input file records, i.e., (1) File 1 record-lighter event time (LRDY) (2) File 2 record-lighter event time (LRDY) (3) File 3 record-forward or retrograde movement (F/R)
NFINC	Logical	-	Input	Equals false on first call and true on additional calls. When NFINC equals false, automatically prints complete record file title.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
TBLUNIT	NUNIT4	None

Subroutines Called

None

MAIN SHPGEN

SUBROUTINE RPTCOM (6)

Purpose

To write a complete record.

Calling Sequence

CALL RPTCOM (NLRDY, NLGTPS, NPAM, NFCOM, LSHPT, MINS)

<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
NLRDY	Integer	6	Input	First six parameters of input file 1 record, i.e., (1) Facility ID (2) Date (3) Shift (4) Lighter type. (5) Lighter ID (6) Lighter cycle
NLGTPS	Integer	-	Input	Seventh parameter of input file 1 record, i.e., lighter position.
NPAM	Integer	3	Input	Last parameter of each of the three input file records, i.e., (1) File 1 record-lighter event time (LRDY) (2) File 2 record-lighter event time (LRDY) (3) File 3 record-forward or retro-grade movement (F/R).
NFCOM	Logical	-	Input	Equals false on first call and true on additional calls. When NFCOM equals false, automatically prints complete record file title.
LSHPT	Integer	-	Input	Lighter at ship time (LRDP-LRDY).
MINS	Integer	-	Input	Minutes elapsed from the beginning of the test.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
TBLUNIT	NUNIT5, NUNIT7	None

Subroutines Called

None

MAIN SHPGEN

SUBROUTINE STOP (7)

Purpose

To determine if end-of-file is in any or in all files. Sets KODE to write incomplete records.

Calling Sequence

CALL STOP (NEND, NSTOP, NSTOP1, KODE)

<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
NEND	Integer	3	Input	Equals EOF when end of file is reached.
NSTOP	Integer	-	Output	When NSTOP is false, all files contain data. When NSTOP is true, at least one file is out of data.
NSTOP1	Logical	-	Output	When NSTOP1 is true, all files are out of data.
KODE	Integer	3	Output	(1) Flag to read a file 1 record. (2) Flag to read a file 2 record. (3) Flag to read a file 3 record.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
None	None	None

Subroutines Called

None

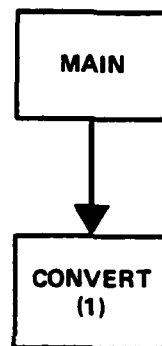


FIGURE 4.5. PREV STRUCTURE

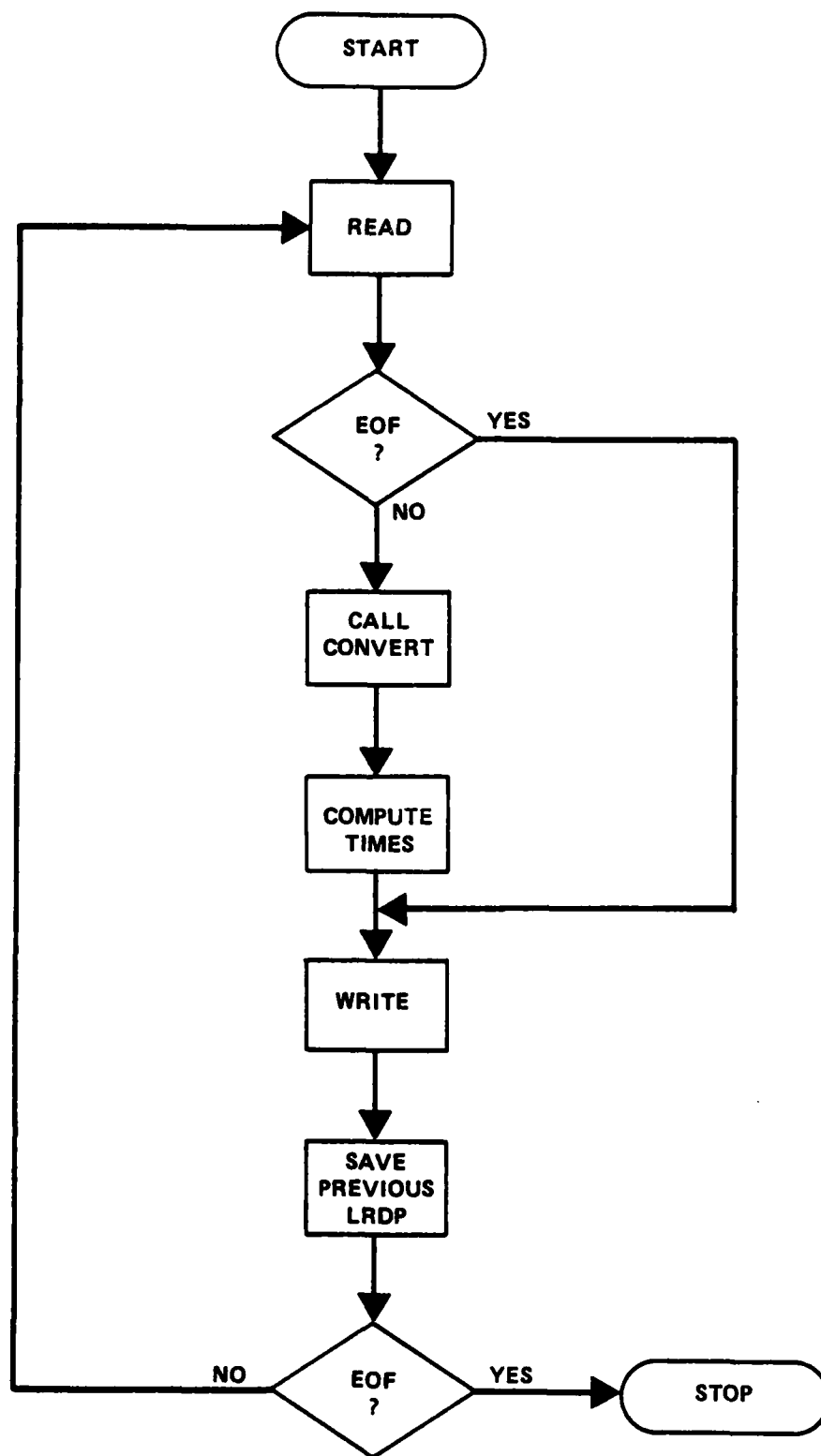


FIGURE 4.6. MAIN PROGRAM (PREV) FLOW DIAGRAM

MAIN PROGRAM PREV

Purpose

To save previous LRDP on present lighter at ship record, and to compute lighter succession time and lighter at ship cycle time.

Program Statement

PROGRAM PREV (TAPE1, OUT, OUTPUT)

<u>FILE NAME</u>	<u>TYPE</u>	<u>USE</u>	<u>DESCRIPTION</u>
TAPE1	Binary	Input	A TAPE1 record contains: (1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Direction (F/R) (9) LRDY (10) LRDP (11) Lighter at ship time (LRDY-LRDP) (12) Minutes elapsed from the beginning of the test.
OUT	Binary	Output	An OUT record contains: (1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Direction (F/R) (9) LRDY (10) LRDP (11) Lighter at ship time (LRDY-LRDP) (12) Lighter succession time (LRDY-previous LRDP) (13) Lighter at ship cycle time (LRDP-previous LRDP)
OUTPUT	BCD	Output	An OUTPUT record contains the same parameters as the OUT record contains.

MAIN PREV

SUBROUTINE CONVERT (1)

Purpose

Converts day, shift, hours, and minutes to minutes from the beginning of the test.

Calling Sequence

CALL CONVERT (NDAY, NSHIFT, NHM, MINS)

<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
NDAY	Integer	-	Input	Julian date.
NSHIFT	Integer	-	Input	Shift 1 or shift 2.
NHM	Integer	-	Input	Time of day in hours and minutes (military time).
MINS	Integer	-	Output	Minutes elapsed from the beginning of the test.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
None	None	None

Subroutines Called

None

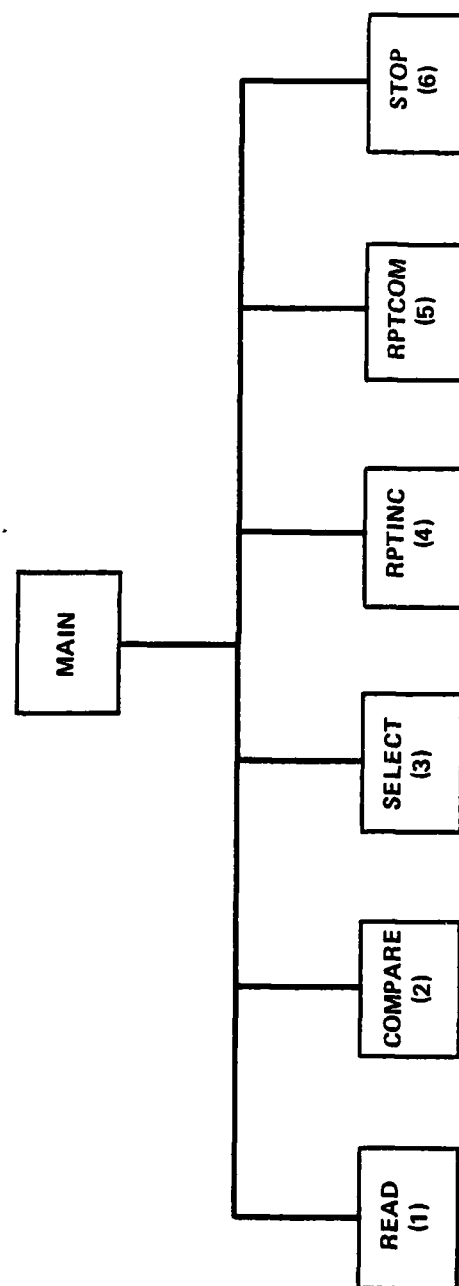


FIGURE 4.7. STEP8 STRUCTURE

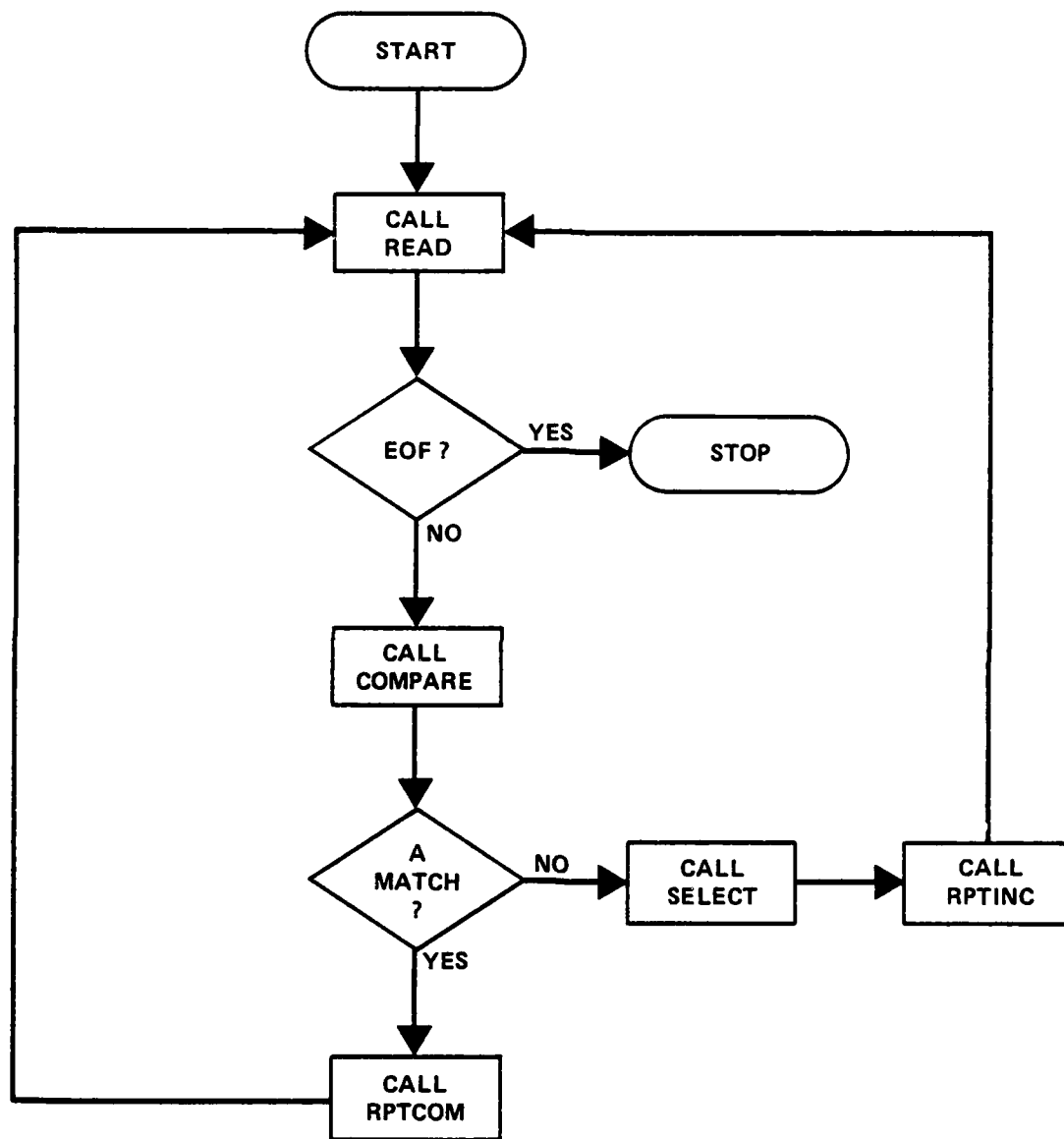


FIGURE 4.8. PROGRAM (STEP8) FLOW DIAGRAM

MAIN PROGRAM STEP8

Purpose

To add cargo type and cargo ID to lighter records.

Program Statement

PROGRAM STEP8 (UNIT4, UNIT7, OUTI7, OUTC7, UNIT8, OUTPUT)

<u>FILE NAME</u>	<u>TYPE</u>	<u>USE</u>	<u>DESCRIPTION</u>
UNIT4	Binary	Input	An UNIT4 record contains: (1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Direction (9) LRDY (10) LRDP (11) Lighter at ship time (LRDY-LRDP) (12) Lighter succession time (LRDY-previous LRDP) (13) Lighter at ship cycle time (LRDP-previous LRDP)
UNIT7	BCD	Input	An UNIT7 record contains: (1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Cargo type (9) Cargo ID
OUTI7	BCD	Output	OUTI7 contains a non-matching UNIT7 or UNIT4 record.
OUTC7	BCD	Output	A OUTC7 record contains: (1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Direction (F/R) (9) LRDY (10) LRDP (11) Lighter at ship time (LRDY-LRDP)

<u>FILE NAME</u>	<u>TYPE</u>	<u>USE</u>	<u>DESCRIPTION</u>
OUTC7 (Cont'd)			(12) Lighter succession time (LRDY-previous LRDP) (13) Lighter at ship cycle time (LRDP-previous LRDP) (14) Cargo type (15) Cargo ID
UNIT8	Binary	Output	An UNIT8 record contains the same paramaters as the above OUTC7 record.
OUTPUT	BCD	Output	Contains systems messages, if any.

MAIN STEP8

SUBROUTINE READ (1)

Purpose

To read a record from the lighter times file and/or the cargo ID file.

Calling Sequence

CALL READ (KODE, LIGHTER, NCARGO, NEND)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
KODE	Integer	2	Input	(1) Flag to read a file 1 record (2) Flag to read a file 2 record
LIGHTER	Integer	13	Output	(1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Direction (9) Lighter ready time (LRDY) (10) Lighter ready to depart time (LRDP) (11) Lighter in position time (LRDP-LRDY) (12) Lighter succession time (LRDY-previous LRDY) (13) Lighter at ship cycle time (LRDP-previous LRDP)
NCARGO	Integer	9	Output	(1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Cargo type (9) Cargo ID
NEND	Integer	2	Output	Equals EOF when end-of-file is reached.

Common Blocks

Block Name

TBLUNIT

Inputs

NUNIT1, NUNIT2,
NUNIT3, NUNIT4,
NUNIT5, NUNIT6

Output

NONE

Subroutines Called

None

MAIN STEP8

SUBROUTINE COMPARE (2)

Purpose

To determine if two records match on six match keys. If no match, it returns to the main program the first key where the match failed.

Calling Sequence

CALL COMPARE (NTEST1, NTEST2, MATCH, NFAIL1, NFAIL2)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
NTEST1	Integer	13	Input	(1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Direction (9) Lighter ready time (LRDY) (10) Lighter ready to depart time (LRDP) (11) Lighter in position time (LRDP-LRDY) (12) Lighter succession time (LRDY-previous LRDY) (13) Lighter at ship cycle time (LRDP-previous LRDP)
NTEST2	Integer	9	Input	(1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Cargo type (9) Cargo ID
MATCH	Logical	-	Output	MATCH equals true if a match was found. MATCH equals false if the match failed.

MAIN STEP8

SUBROUTINE COMPARE (2) (CONTINUED)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
NFAIL1	Integer		Output	Contains the first LIGHTER match key where the match failed.
NFAIL2	Integer		Output	Contains the first NCARGO match key where the match failed.

Common Blocks

Block Name

Input

Output

NONE

NONE

Subroutines Called

NONE

MAIN STEP8

SUBROUTINE SELECT (3)

Purpose

To determine which file to read

Calling Sequence

CALL SELECT (NFAIL1, NFAIL2, KODE)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
NFAIL1	Integer	-	Input	Contains the first LIGHTER match key where the match failed.
NFAIL2	Integer	-	Input	Contains the first NCARGO match key where the match failed.
KODE	Integer	2	Output	(1) Flag to read a file 1 record. (2) Flag to read a file 2 record.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
NONE	NONE	NONE

Subroutines Called

NONE

MAIN STEP 8

SUBROUTINE RPTINC (4)

Purpose

To write an incomplete record

Calling Sequence

CALL RPTINC (KODE, LIGHTER, NCARGO, NPRINT)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
KODE	Integer	2	Input	(1) Flag to read a file 1 record. (2) Flag to read a file 2 record.
LIGHTER	Integer	13	Input	(1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Direction (9) Lighter ready time (LRDY) (10) Lighter ready to depart time (LRDP) (11) Lighter in position time (LRDP-LRDY) (12) Lighter succession time (LRDY-previous LRDY) (13) Lighter at ship cycle time (LRDP-previous LRDP)
NCARGO	Integer	9	Input	(1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Cargo type (9) Cargo ID
NPRINT	Logical	-	Input/ Output	If NPRINT equals false writes an incomplete lighter record; if true writes an incomplete cargo record.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
TBLUNIT	NUNIT1, NUNIT2, NUNIT3, NUNIT4, NUNIT5, NUNIT6	NONE

Subroutines Called

NONE

MAIN STEP 8

SUBROUTINE RPTCOM (5)

Purpose

To write a complete record

Calling Sequence

CALL RPTCOM (LIGHTER, NCARGO, NPRINT)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
LIGHTER	Integer	13	Input	(1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Direction (9) Lighter ready time (LRDY) (10) Lighter ready to depart time (LRDP) (11) Lighter in position time (LRDP-LRDY) (12) Lighter succession time (LRDY-previous LRDY) (13) Lighter at ship cycle time (LRDP-previous LRDP)
NCARGO	Integer	9	Input	(1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Cargo type (9) Cargo ID
NPRINT	Logical	-	Input/ Output	If NPRINT equals false writes a complete record; if true, the present lighter record is complete.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
NBLUNIT	NUNIT1, NUNIT2 NUNIT3, NUNIT4, NUNIT5, NUNIT6	NONE

Subroutines Called NONE

MAIN STEP 8

SUBROUTINE STOP (6)

Purpose

To determine if EOF in any file or if EOF in all files. Sets KODE to indicate which records to read.

Calling Sequence

CALL STOP (NEND, NSTOP, NSTOP1, KODE)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
NEND	Integer	2	Input	Equals EOF when end of file is reached.
NSTOP	Logical	-	Output	When NSTOP is false, all files contain data. When NSTOP is true, at least one file is out of data.
NSTOP1	Logical	-	Output	When NSTOP1 is true all files are out of data.
KODE	Integer	2	Output	(1) Flag to read a file 1 record. (2) Flag to read a file 2 record.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
NONE	NONE	NONE

Subroutines Called

NONE

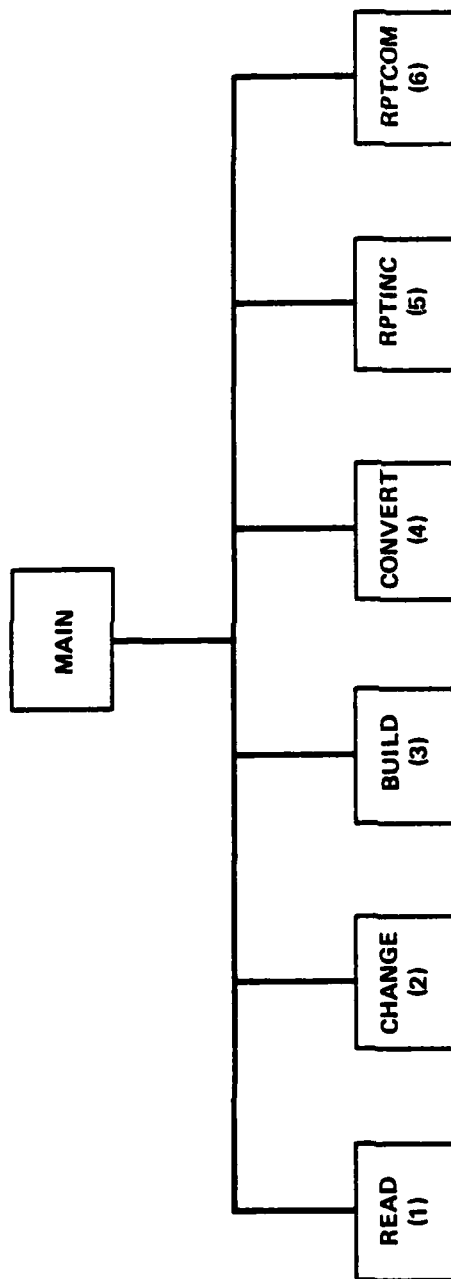


FIGURE 4.9. CARGO ID STRUCTURE

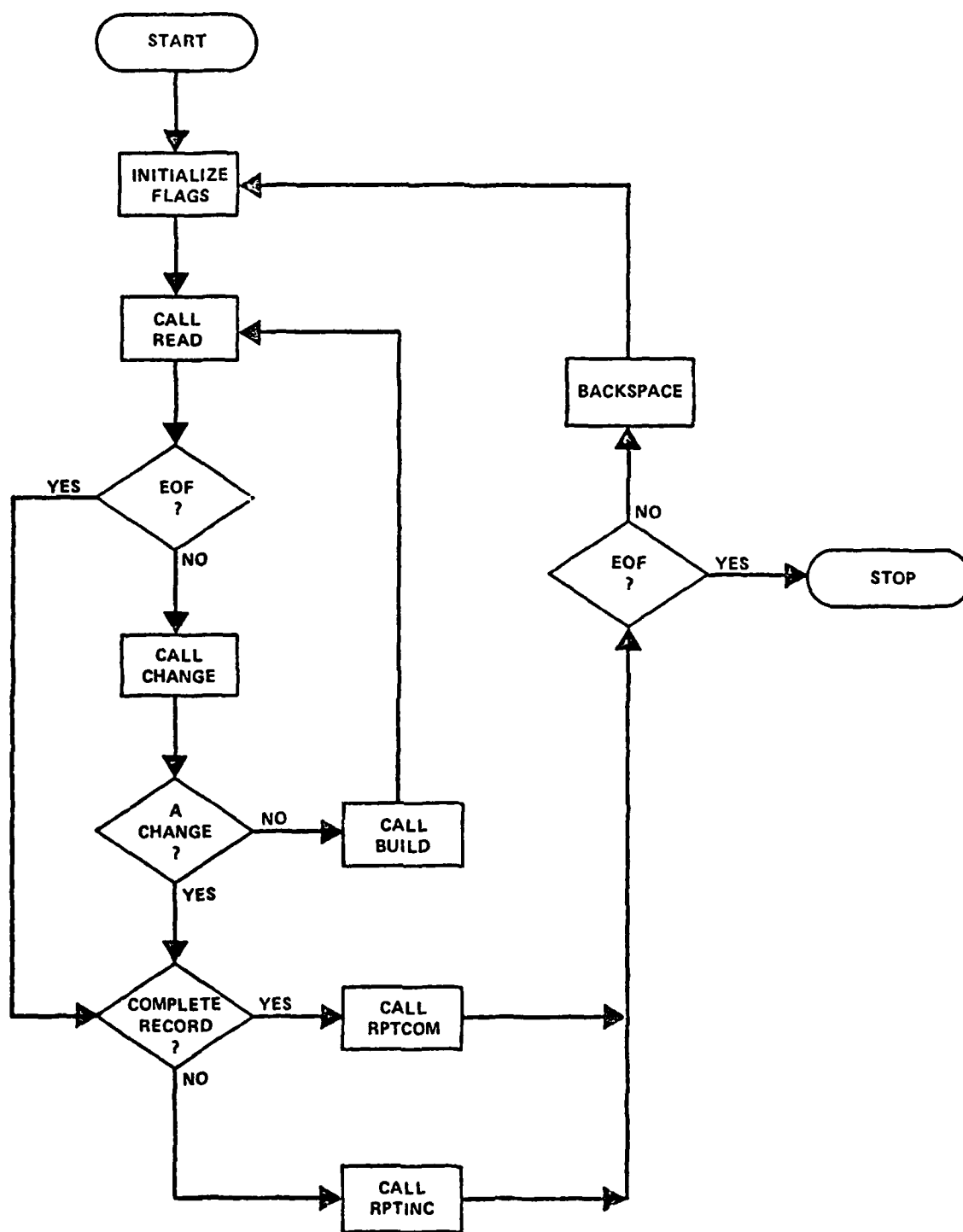


FIGURE 4.10. MAIN PROGRAM (CARGO ID) FLOW DIAGRAM

MAIN PROGRAM CARGOID

Purpose

To construct one record which contains LDOC, LDLK, LIFT, CPSN, LAND, LDUN, Delay start and delay type.

Program Statement

PROGRAM CARGOID (CARGO, OUTI1, OUTC1, UNIT11, OUTPUT)

<u>FILE NAME</u>	<u>TYPE</u>	<u>USE</u>	<u>DESCRIPTION</u>
CARGO	BCD	Input	A CARGO record contains: (1) Facility ID (2) Julian date (3) Shift (4) Operator ID (5) Lighter type (6) Lighter ID (7) Cargo ID (8) Cargo cycle (9) Cargo event name (10) Cargo event time (11) Delay type (12) Delay category (13) Delay duration
OUTI1	BCD	Output	Outputs records that are missing one or more of the following: LDOC, LDLK, LIFT, CPSN, LAND, LDUN.
OUTC1	BCD	Output	An OUTC1 record contains: (1) Facility ID (2) Julian date (3) Shift (4) Operator ID (5) Lighter type (6) Lighter ID (7) Cargo ID (8) Cargo cycle (9) LDOC (10) LDLK (11) LIFT (12) CPSN (13) LAND (14) LDUN (15) Delay start (16) Delay type (17) Delay sequence number (18) Minutes elapsed from the start of the test.

MAIN PROGRAM CARGOID (CONTINUED)

<u>FILE NAME</u>	<u>TYPE</u>	<u>USE</u>	<u>DESCRIPTION</u>
UNIT11	Binary	Output	An UNIT11 record contains the same parameters as the above OUTC1 record.
OUTPUT	BCD	Output	Contains systems messages, if any.

MAIN CARGOID

SUBROUTINE READ (1)

Purpose

To read one input record.

Calling Sequence

CALL READ (NDATA, STOP)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
NDATA	Integer	13	Output	(1) Facility ID (2) Julian date (3) Shift (4) Operator ID (5) Lighter type (6) Lighter ID (7) Cargo ID (8) Cargo cycle (9) Cargo event name (10) Cargo event time (11) Delay type (12) Delay category (13) Delay duration
STOP	Logical	-	Output	When STOP equals true, at least one file is out of data. When STOP equals false, all files contain data.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
TBLUNIT	NUNIT1, NUNIT2 NUNIT3, NUNIT4	NONE

Subroutines Called

None

MAIN CARGOID

SUBROUTINE CHANGE (2)

Purpose

To set the value of END to true when the facility ID, Lighter type, Lighter ID, cargo ID, or cargo cycle change.

Calling Sequence

CALL CHANGE (NDATA, FIRST, END)

VARIABLE	TYPE	DIMENSION	USE	DESCRIPTION
NDATA	Integer	13	Input	(1) Facility ID (2) Julian data (3) Shift (4) Operator ID (5) Lighter type (6) Lighter ID (7) Cargo ID (8) Cargo cycle (9) Cargo event name (10) Cargo event time (11) Delay type (12) Delay category (13) Delay duration
FIRST	Logical	-	Input/ Output	Equals true on first call for each cargo ID. Equals false on additional calls for the same cargo ID.
END	Logical	-	Output	Flag to indicate a change in facility ID, Lighter type, Lighter ID; cargo ID or cargo cycle change.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
NONE	NONE	NONE

Subroutines Called

NONE

MAIN CARGOID

SUBROUTINE BUILD (3)

Purpose

To construct a record which contains the LDOC, LDLK, LIFT, CPSN, LAND, and the DUN. Also constructs NDELAY record(s) which contain delay time, type, and duration. Sets a flag to indicate if the record is complete.

Calling Sequence

CALL BUILD (NDATA, FIRST, NPAM, NDELAY1, NDELAY2, NSEQ, FLAG, NSAVE)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
NDATA	Integer	13	Input	(1) Facility ID (2) Julian data (3) Shift (4) Operator ID (5) Lighter type (6) Lighter ID (7) Cargo ID (8) Cargo cycle (9) Cargo event name (10) Cargo event time (11) Delay type (12) Delay category (13) Delay duration
FIRST	Logical	-	Input/ Output	Equals true on first call in each cargo ID, Equal, false on additional calls for the same cargo ID.
NPAM	Integer	6	Output	(1) Cargo event time (LDOC) (2) Cargo event time (LDLK) (3) Cargo event time (LIFT) (4) Cargo event time (CPSN) (5) Cargo event time (LAND) (6) Cargo event time (LDUN)
NDELAY1	Integer	20	Output	Delay event time.
NDELAY2	Integer	20	Output	Delay event type.
NSEQ	Integer	-	Output	Number of delays.
FLAG	Logical	-	Output	Equals true when a record is complete. Equals false when a record is incomplete.

MAIN CARGOID

SUBROUTINE BUILD (3) CONTINUED

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
NSAVE	Integer	8	Output	(1) Facility ID (2) Julian data (3) Shift (4) Operator ID (5) Lighter type (6) Lighter ID (7) Cargo ID (8) Cargo cycle

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
NONE	NONE	NONE

Subroutines Called

NONE

MAIN CARGOID

SUBROUTINE CONVERT (4)

Purpose

Converts days, shifts, hours and minutes to minutes from the beginning of the test.

Calling Sequence

CALL CONVERT (NDAY, NSHIFT, NHM, MINS)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
NDAY	Integer	-	Input	Julian date
NSHIFT	Integer	-	Input	Shift 1 or shift 2
NHM	Integer	-	Input	Time of day in hours and minutes (military time).
MINS	Integer	-	Output	Minutes elapsed from the beginning of the test.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
NONE	NONE	NONE

Subroutines Called

NONE

MAIN CARGOID

SUBROUTINE RPTINC (5)

Purpose

To write an incomplete record.

Calling Sequence

CALL RPTINC (NDATA, NPAM, NSEQ, NDELAY1, NDELAY2)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
NDATA	Integer	8	Input	(1) Facility ID (2) Julian date (3) Shift (4) Operator ID (5) Lighter type (6) Lighter ID (7) Cargo ID (8) Cargo cycle
NPAM	Integer	6	Input	(1) Cargo event time (LDOC) (2) Cargo event time (LDLK) (3) Cargo event time (LIFT) (4) Cargo event time (CPSN) (5) Cargo event time (LAND) (6) Cargo event time (LDUN)
NSEQ	Integer	-	Input	Number of Delays.
NDELAY1	Integer	20	Input	Delay event time.
NDELAY2	Integer	20	Input	Delay event type.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
TBLUNIT	NUNIT1, NUNIT2 NUNIT3, NUNIT4	NONE

Subroutines Called

NONE

MAIN CARGOID

SUBROUTINE RPTCOM (6)

Purpose

To write a complete record.

Calling Sequence

CALL RPTCOM (NDATA, NPAM, NSEQ, NDELAY1, NDELAY2, NTIME)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
NDATA	Integer	8	Input	(1) Facility ID (2) Julian date (3) Shift (4) Operator ID (5) Lighter type (6) Lighter ID (7) Cargo ID (8) Cargo cycle
NPAM	Integer	6	Input	(1) Cargo event time (LDOC) (2) Cargo event time (LDLK) (3) Cargo event time (LIFT) (4) Cargo event time (CPSN) (5) Cargo event time (LAND) (6) Cargo event time (LDUN)
NSEQ	Integer	-	Input	Number of delays.
NDELAY1	Integer	20	Input	Delay event time.
NDELAY2	Integer	20	Input	Delay event type.
NTIME	Integer	-	Input	Elapsed time from the beginning of the test.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
TBLUNIT	NUNIT1, NUNIT2, NUNIT3, NUNIT4	NONE

Subroutines Called

NONE

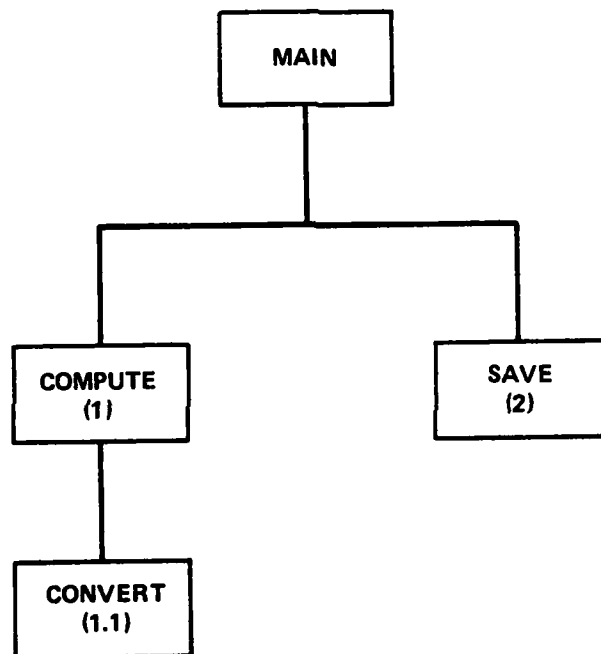


FIGURE 4.11. STEP 13 STRUCTURE

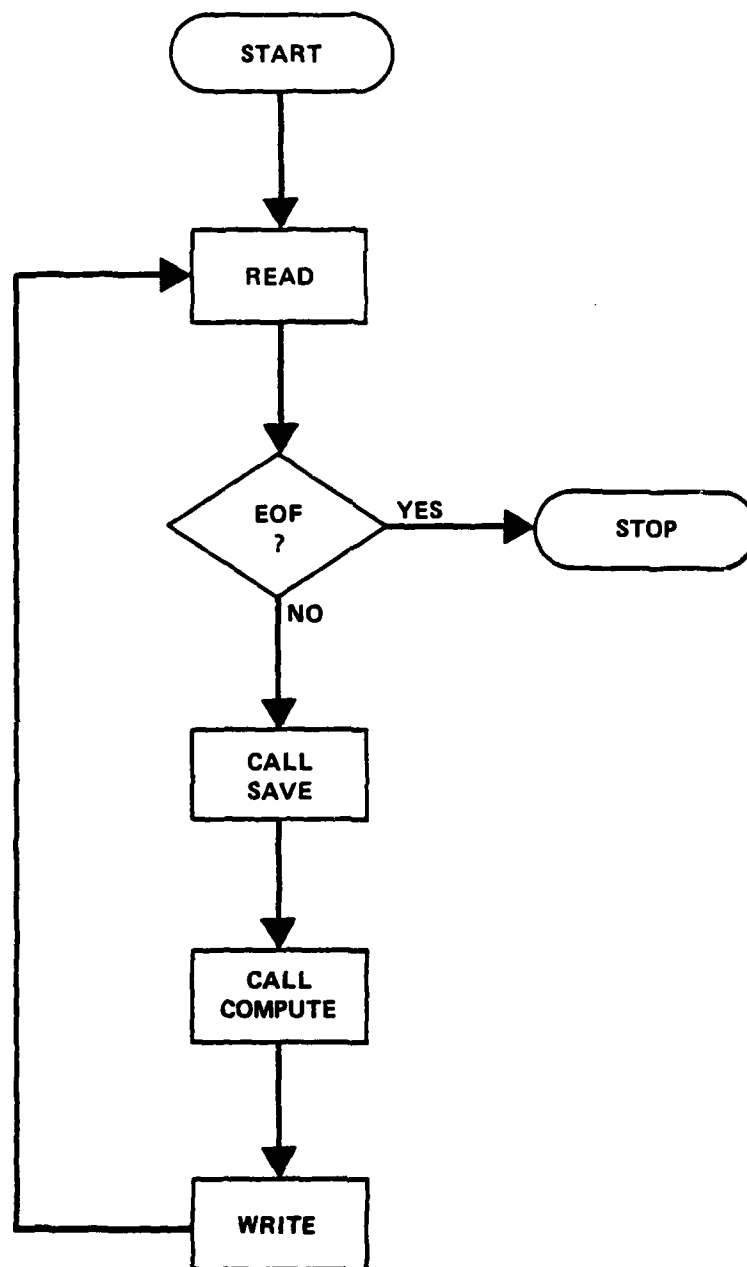


FIGURE 4.12. MAIN PROGRAM (STEP 13) FLOW DIAGRAM

MAIN PROGRAM STEP13

Purpose

To write previous LDUN on present record and to compute the parts of the crane cycle.

Program Statement

PROGRAM (TAPE1, TAPE2, TAPE3, OUTPUT)

<u>FILE NAME</u>	<u>TYPE</u>	<u>USE</u>	<u>DESCRIPTION</u>
TAPE1	Binary	Input	(1) Facility ID (2) Julian date (3) Shift (4) Operator ID (5) Lighter type (6) Lighter ID (7) Cargo ID (8) Cargo cycle (9) LDOC (10) LDLK (11) LIFT (12) CPSN (13) LAND (14) LDUN (15) Delay start (16) Delay type (17) Sequence number (18) Minutes elapsed from the beginning of the test.
TAPE2	Binary	Output	A TAPE2 record contains: (1) Facility ID (2) Julian date (3) Shift (4) Operator ID (5) Lighter type (6) Lighter ID (7) Cargo ID (8) Cargo cycle (9) LDOC (10) LDOC - previous LDUN (11) LDLK (12) LDLK - LDOC (13) LIFT (14) LIFT - LDLK (15) CPSN (16) CPSN - LIFT (17) LAND (18) LAND - CPSN

MAIN PROGRAM STEP13 (CONTINUED)

<u>FILE NAME</u>	<u>TYPE</u>	<u>USE</u>	<u>DESCRIPTION</u>
			(19) LDUN
			(20) LDUN - LAND
			(21) Delay start
			(22) Delay type
			(23) Sequence number
TAPE3	BCD	Output	An OUTPUT record contains the same parameters as a TAPE2 record.
OUTPUT	BCD	Output	Contains system messages, if any.

MAIN STEP13

SUBROUTINE COMPUTE (1)

Purpose

To compute the component parts of a crane cycle.

Calling Sequence

CALL COMPUTE (NDATA, NPREV, NTIME)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
NDATA	Integer	17	Input	A NDATA record contains: (1) Facility ID (2) Julian date (3) Shift (4) Operator ID (5) Lighter type (6) Lighter ID (7) Cargo ID (8) Cargo cycle (9) LDOC (10) LDLK (11) LIFT (12) CPSN (13) LAND (14) LDUN (15) Delay start (16) Delay type (17) Sequence number
NPREV	Integer	3	Input	A NPREV record contains: (1) previous Julian date (2) previous shift (3) previous time
NTIME	Integer	6	Output	A NTIME record contains: (1) LDOC - previous LDUN (2) LDLK - LDOC (3) LIFT - LDLK (4) CPSN - LIFT (5) LAND - CPSN (6) LDUN - LAND

MAIN STEP13

SUBROUTINE SAVE

Purpose

To return previous LDUN.

Calling Sequence

CALL SAVE (NFACID, NDATE, NSHIFT, NLDUN, NPREV)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
NFACID	Integer	-	Input	Facility ID
NDATE	Integer	-	Input	Julian date
NSHIFT	Integer	-	Input	Shift
NLDUN	Integer	-	Input	Present LDUN
NPREV	Integer	-	Input	(1) previous Julian date (2) previous shift (3) previous LDUN

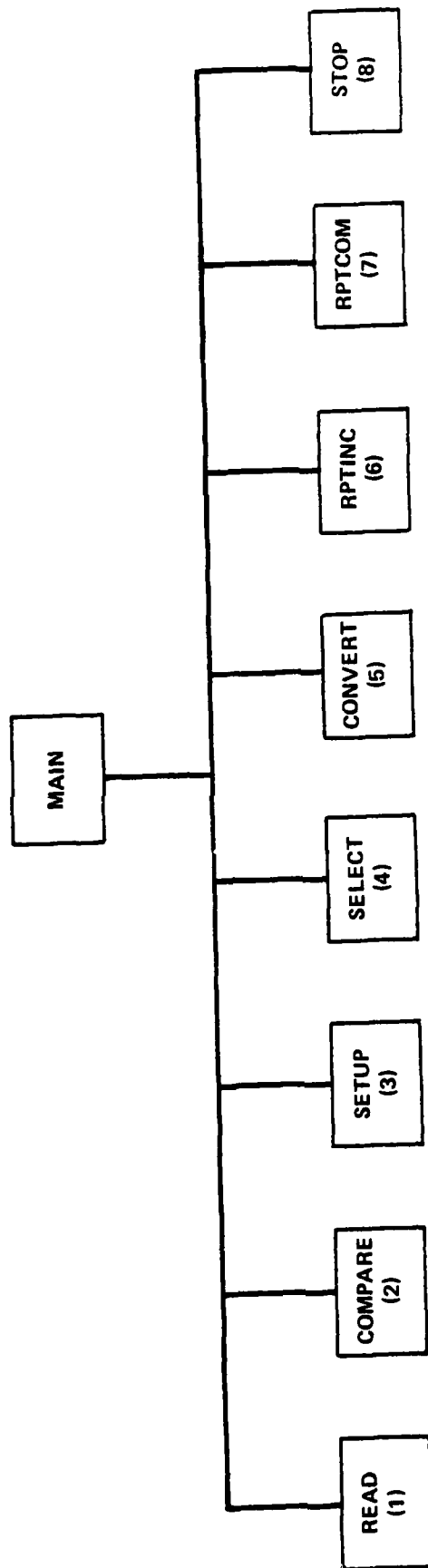


FIGURE 4.13. STEP 15 STRUCTURE

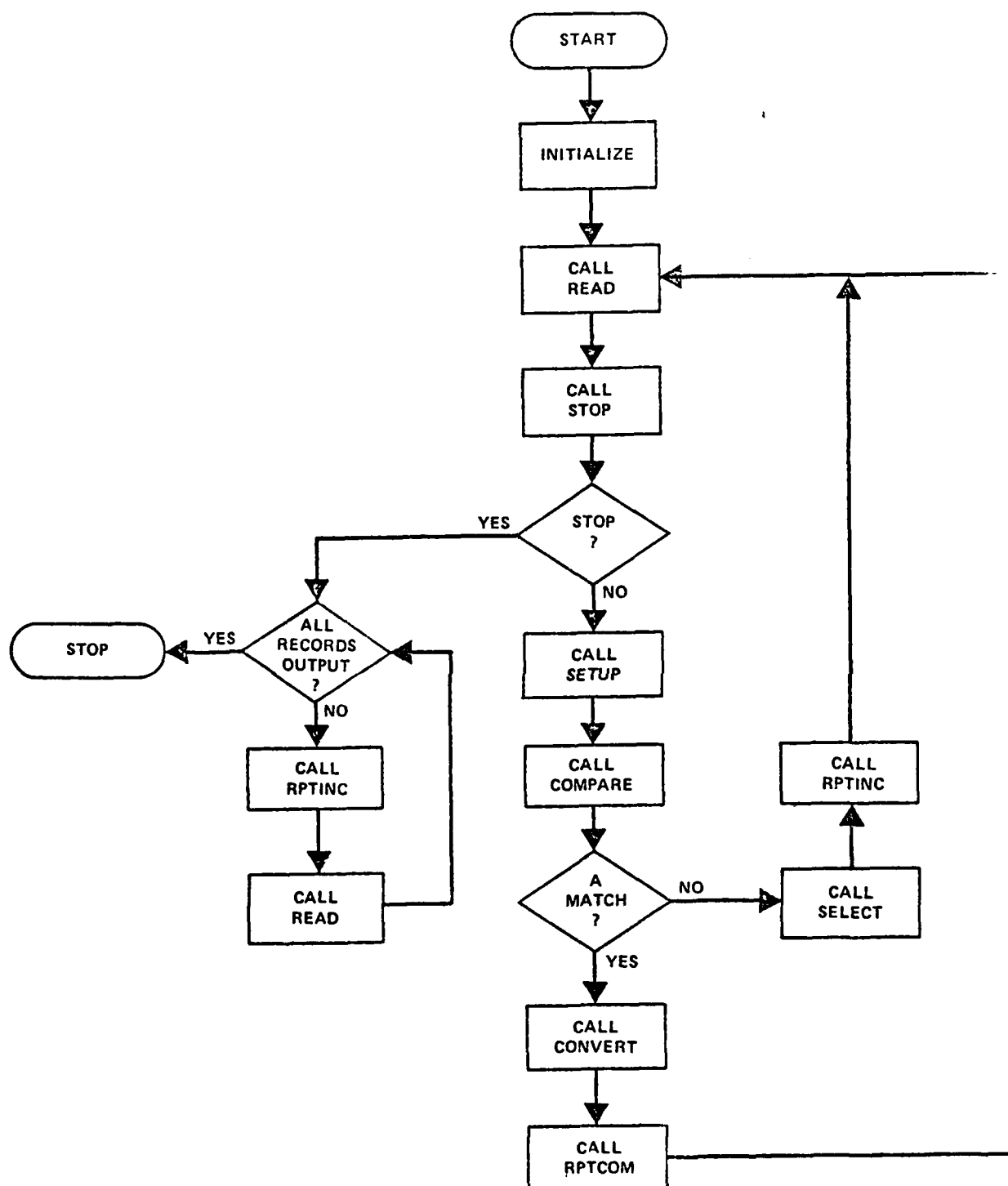


FIGURE 4.14. MAIN PROGRAM (STEP 15) FLOW DIAGRAM

MAIN PROGRAM STEP15

Purpose

To combine a lighter record and a cargo record into one record.

Program Statement

PROGRAM STEP15 (UNIT8A, UNIT14, OUT115, OUTC16, UNIT15, OUTPUT)

<u>FILE NAME</u>	<u>TYPE</u>	<u>USE</u>	<u>DESCRIPTION</u>
UNIT8A	Binary	Input	<p>A UNIT8A record contains:</p> <ol style="list-style-type: none"> (1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Direction (F/R) (9) LRDY (10) LRDP (11) Lighter at ship time (LRDY-LRDP) (12) Lighter succession time (LRDY-previous LRDP) (13) Lighter at ship cycle time (LRDP-previous LRDP) (14) Cargo type (15) Cargo ID
UNIT14	Binary	Input	<p>A UNIT14 record contains:</p> <ol style="list-style-type: none"> (1) Facility ID (2) Julian date (3) Shift (4) Operator ID (5) Lighter type (6) Lighter ID (7) Cargo ID (8) Cargo cycle (9) LDOC (10) LDOC - previous LDUN (11) LDLK (12) LDLK - LDOC (13) LIFT (14) LIFT - LDLK (15) CPSN (16) CPSN - LIFT (17) LAND (18) LAND - CPSN (19) LDUN (20) LDUN - LAND (21) Delay start (22) Delay type (23) Delay sequence number

MAIN PROGRAM STEP15 (CONTINUED)

<u>FILE NAME</u>	<u>TYPE</u>	<u>USE</u>	<u>DESCRIPTION</u>
OUTI15	BCD	Output	Contains a non-matching UNIT8A or UNIT14 record.
OUTC15	BCD	Output	An OUTC15 record contains: (1) Facility ID (2) Julian date (3) shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Direction (F/R) (9) LRDY (10) LRDP (11) LRDP - LRDY (12) LRDY - previous LRDP (13) LRDP - previous LRDP (14) Cargo type (15) Cargo ID (16) Operator ID (17) LDOC (18) LDOC - previous LDUN (19) LDLK (20) LDLK - LDOC (21) LIFT (22) LIFT - LDLK (23) CPSN (24) CPSN - LIFT (25) LAND (26) LAND - CPSN (27) LDUN (28) LDUN - LAND (29) Delay start (30) Delay type (31) Delay sequence number (32) Minutes elapsed from the beginning of the test.
UNIT15	Binary	Output	An UNIT15 record contains the same parameters as an OUTC15 record.
OUTPUT	BCD	Output	Contains systems messages, if any.

MAIN STEP15

SUBROUTINE READ (1)

Purpose

To read a record from the lighter times file and/or the cargo times file.

Calling Sequence

CALL READ (KODE, LIGHTER, NCARGO, NEND)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
KODE	Integer	2	Input	(1) Flag to read a file 1 record. (2) Flag to read a file 2 record.
LIGHTER	Integer	15	Output	(1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Direction (9) Lighter event time (LRDY) (10) Lighter event time (LRDP) (11) LRDP - LRDY (12) LRDY - previous LRDP (13) LRDP - previous LRDP (14) Cargo type (15) Cargo ID
NCARGO	Integer	23	Output	(1) Facility ID (2) Julian date (3) Shift (4) Operator ID (5) Lighter type (6) Lighter ID (7) Cargo ID (8) Cargo cycle (9) Cargo event time (LDOC) (10) LDOC - previous LDUN (11) Cargo event time (LDLK) (12) LDLK - LDOC (13) Cargo event time (LIFT) (14) LIFT - LDLK (15) Cargo event time (CPSN) (16) CPSN - LIFT (17) Cargo event time (LAND) (18) LAND - CPSN (19) Cargo event time (LDUN) (20) LDUN - LAND (21) Delay start time (22) Delay type (23) Delay sequence number

MAIN STEP15 (CONTINUED)

SUBROUTINE READ (1)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
NEND	Integer	2	Output	Equals EOF when end of file is reached.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
TBLUNIT	NUNIT1, NUNIT2, NUNIT3, NUNIT4, NUNIT5, NUNIT6	NONE

Subroutines Called

NONE

MAIN STEP15

SUBROUTINE COMPARE (2)

Purpose

To determine if two records match on four match keys (i.e., facility ID, Lighter type, lighter ID and cargo ID). If there is no match, it returns to the main program the first key where the match failed.

Calling Sequence

CALL COMPARE (NTEST1, NTEST2, MATCH, NFAIL1, NFAIL2)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
NTEST1	Integer	4	Input	NTEST1 is constructed from the LIGHTER record (1) Facility ID (2) Lighter type (3) Lighter ID (4) Cargo ID
NTEST2	Integer	4	Input	NTEST2 is constructed from the NCARGO record (1) Facility ID (2) Lighter type (3) Lighter ID (4) Cargo ID
MATCH	Logical	-	Output	MATCH equals true if a match was found. MATCH equals false if the match failed.
NFAIL1	Integer	-	Output	Contains the first LIGHTER match key where the match failed.
NFAIL2	Integer	-	Output	Contains the first NCARGO match key where the match failed.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
NONE	NONE	NONE

Subroutines Called

NONE

MAIN STEP15

SUBROUTINE SETUP (3)

Purpose

To construct two arrays with facility ID, lighter type, lighter ID, and cargo ID.

Calling Sequence

CALL SETUP (LIGHTER, NCARGO, NTEST1, NTEST2)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
LIGHTER	Integer	15	Input	(1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Direction (9) Lighter event time (LRDY) (10) Lighter event time (LRDP) (11) LRDP - LRDY (12) LRDY - previous LRDP (13) LRDP - previous LRDP (14) Cargo type (15) Cargo ID
NCARGO	Integer	23	Input	(1) Facility ID (2) Julian date (3) Shift (4) Operator ID (5) Lighter type (6) Lighter ID (7) Cargo ID (8) Cargo cycle (9) Cargo event time (LDOC) (10) LDOC - previous LDUN (11) Cargo event time (LDLK) (12) LDLK - LDOC (13) Cargo event time (LIFT) (14) LIFT - LDLK (15) Cargo event time (CPSN) (16) CPSN - LIFT (17) Cargo event time (LAND) (18) LAND - CPSN (19) Cargo event time (LDUN) (20) LDUN - LAND (21) Delay start time (22) Delay type (23) Delay sequence number

MAIN STEP15

SUBROUTINE SETUP (3) (CONTINUED)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
NTEST1	Integer	4	Output	NTEST1 is constructed from the LIGHTER record. (1) Facility ID (2) Lighter type (3) Lighter ID (4) Cargo ID
NTEST2	Integer	4	Output	NTEST2 is constructed from the NCARGO record. (1) Facility ID (2) Lighter type (3) Lighter ID (4) Cargo ID

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
NONE	NONE	NONE

Subroutines Called

NONE

MAIN STEP15

SUBROUTINE SELECT (4)

Purpose

To determine which of two input files to read.

Calling Sequence

CALL SELECT (NFAIL1, NFAIL2, KODE)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
NFAIL1	Integer	-	Output	Contains the first LIGHTER match key where the match failed.
NFAIL2	Integer	-	Output	Contains the first NCARGO match key where the match failed.
KODE	Integer	2	Input	(1) Flag to read a file 1 record. (2) Flag to read a file 2 record.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
NONE	NONE	NONE

Subroutines Called

NONE

MAIN STEP15

SUBROUTINE CONVERT (5)

Purpose

To convert days, shifts, hours, and minutes to minutes from beginning of test.

Calling Sequence

CALL CONVERT (NDAY, NSHIFT, NHM, MINS)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
NDAY	Integer	-	Input	Julian date.
NSHIFT	Integer	-	Input	First or second shift.
NHM	Integer	-	Input	Time of day in hours and minutes (military time).
MINS	Integer	-	Output	Minutes elapsed from the beginning of the test.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
NONE	NONE	NONE

Subroutines Called

NONE

MAIN STEP15

SUBROUTINE RPTINC (6)

Purpose

To write an incomplete lighter or cargo record.

Calling Sequence

CALL RPTINC (KODE, LIGHTER, NCARGO, NPRINT)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
KODE	Integer	2	Input	(1) Flag to read a file 1 record. (2) Flag to read a file 2 record.
LIGHTER	Integer	15	Input	(1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Direction (9) Lighter event time (LRDY) (10) Lighter event time (LRDP) (11) LRDP - LRDY (12) LRDY - previous LRDP (13) LRDP - previous LRDP (14) Cargo type (15) Cargo ID
NCARGO	Integer	23	Input	(1) Facility ID (2) Julian date (3) Shift (4) Operator ID (5) Lighter type (6) Lighter ID (7) Cargo ID (8) Cargo cycle (9) Cargo event time (LDOC) (10) LDOC - previous LDUN (11) Cargo event time (LDLK) (12) LDLK - LDOC (13) Cargo event time (LIFT) (14) LIFT - LDLK (15) Cargo event time (CPSN) (16) CPSN - LIFT (17) Cargo event time (LAND) (18) LAND - CPSN (19) Cargo event time (LDUN) (20) LDUN - LAND (21) Delay start time (22) Delay type (23) Delay sequence number

MAIN STEP15

SUBROUTINE RPTINC (6) (CONTINUED)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
NPRINT	Logical	-	Input	NPRINT equals true when a match was found for a lighter record. NPRINT equals false when a match was not found for a lighter record.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
TBLUNIT	NUNIT1, NUNIT2, NUNIT3, NUNIT4, NUNIT5, NUNIT6	NONE

Subroutines Called

None

MAIN STEP15

SUBROUTINE RPTCOM (7)

Purpose

To write a complete lighter or cargo record.

Calling Sequence

CALL RPTCOM (LIGHTER, NCARGO, NTIME, NPRINT)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
LIGHTER	Integer	15	Input	(1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Direction (9) Lighter event time (LRDY) (10) Lighter event time (LRDP) (11) LRDP - LRDY (12) LRDY - previous LRDP (13) LRDP - previous LRDP (14) Cargo type (15) Cargo ID
NCARGO	Integer	23	Input	(1) Facility ID (2) Julian date (3) Shift (4) Operator ID (5) Lighter type (6) Lighter ID (7) Cargo ID (8) Cargo cycle (9) Cargo event time (LDOC) (10) LDOC - previous LDUN (11) Cargo event time (LDLK) (12) LDLK - LDOC (13) Cargo event time (LIFT) (14) LIFT - LDLK (15) Cargo event time (CPSN) (16) CPSN - LIFT (17) Cargo event time (LAND) (18) LAND - CPSN (19) Cargo event time (LDUN) (20) LDUN - LAND (21) Delay start time (22) Delay type (23) Delay sequence number

MAIN STEP15

SUBROUTINE RPTCOM (7) (CONTINUED)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
NTIME	Integer	-	Input	Elapsed time from the beginning of the test.
NPRINT	Logical	-	Input	NPRINT equals true when a match was found for a lighter record. NPRINT equals false when a match was not found for a lighter record.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
TBLUNIT	NUNIT1, NUNIT2 NUNIT3, NUNIT4, NUNIT5, NUNIT6	NONE

Subroutines Called

NONE

MAIN STEP15

SUBROUTINE STOP (8)

Purpose

To determine if EOF in any file or in all files. Sets KODE to indicate record.

Calling Sequence

CALL STOP (NEND, NSTOP, NSTOP1, KODE)

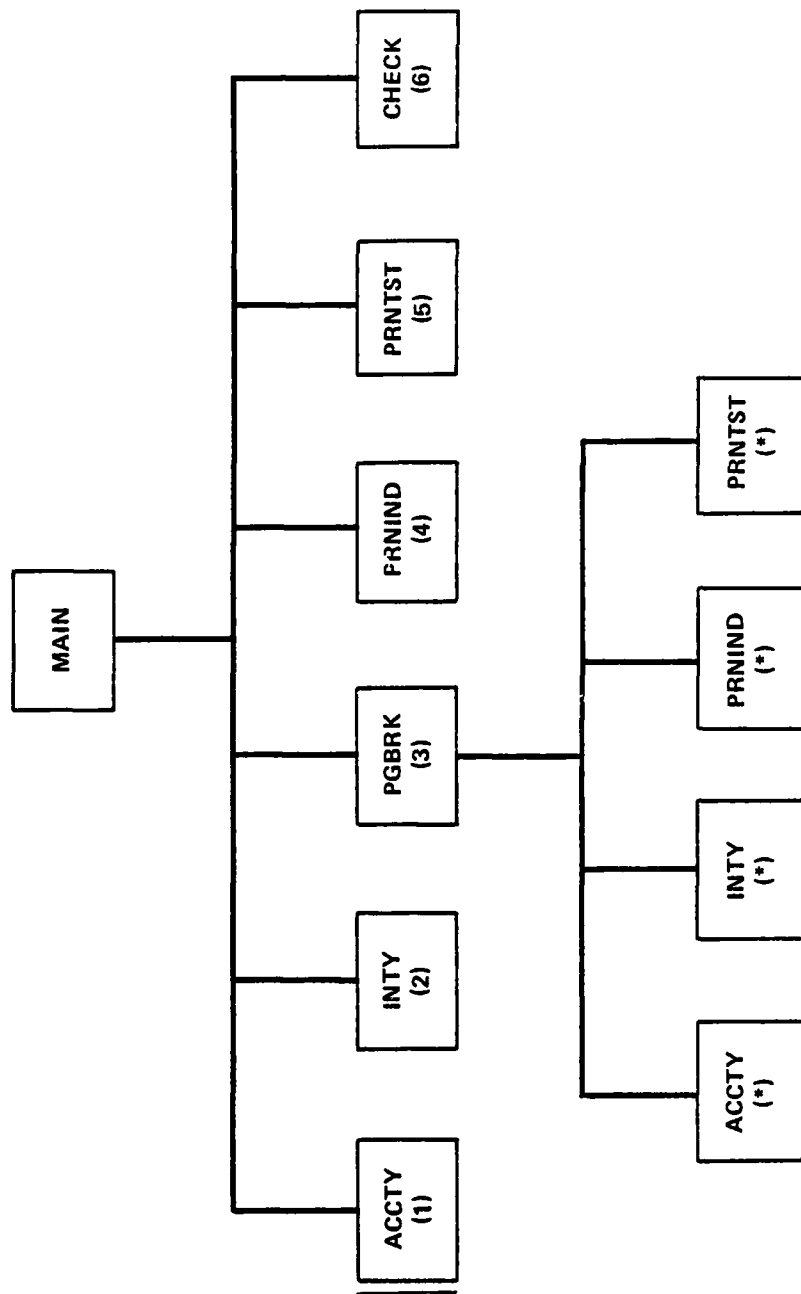
<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
NEND	Integer	2	Input	Equals EOF when end of file is reached.
NSTOP	Logical	-	Input	When NSTOP is false, all files contain data. When NSTOP is true, at least one stop is out of data.
NSTOP1	Logical	-	Input	When NSTOP1 is true, all files are out of data.
KODE	Integer	2	Output	(1) Flag to read a file 1 record. (2) Flag to read a file 2 record.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
NONE	NONE	NONE

Subroutines Called

NONE



* THESE SUBROUTINES ARE ALSO CALLED BY MAIN.

FIGURE 4.15. WRTPT STRUCTURE

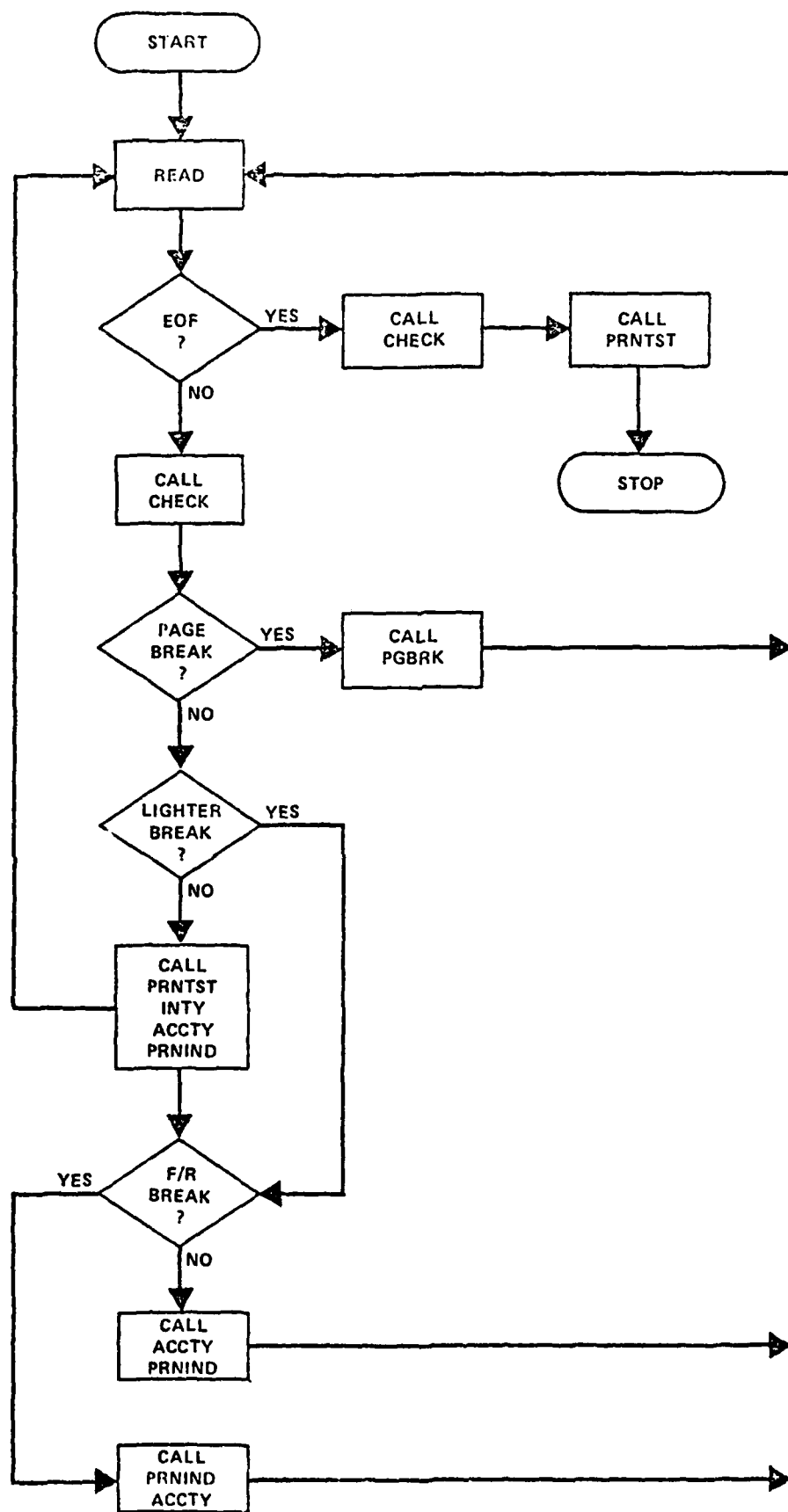


FIGURE 4.16. MAIN PROGRAM (WRTRPT) FLOW DIAGRAM

MAIN PROGRAM WRTRPT

Purpose

To read the sorted lighter and cargo times file, compute statistics and write a report. For each facility, date and shift, statistics are accumulated on lighter types.

Program Statement

PROGRAM WRTRPT (TAPE1, OUTPUT)

<u>FILE NAME</u>	<u>TYPE</u>	<u>USE</u>	<u>DESCRIPTION</u>
TAPE1	Binary	Input	(1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter Position (8) Direction (F/R) (9) LRDY (10) LRDP (11) LRDP - LRDY (12) LRDY - previous LRDP (13) LRDP - previous LRDP (14) Cargo type (15) Cargo ID (16) Operator ID (17) LDOC (18) LDOC - previous LDUN (19) LDLK (20) LDLK - LDOC (21) LIFT (22) LIFT - LDLK (23) CPSN (24) CPSN - LIFT (25) LAND (26) LAND - CPSN (27) LDUN (28) LDUN - LAND (29) Delay start (30) Delay type (31) Delay sequence number (32) Minutes elapsed from the beginning of the test.

MAIN PROGRAM WRTRPT (CONTINUED)

<u>FILE NAME</u>	<u>TYPE</u>	<u>USE</u>	<u>DESCRIPTION</u>
OUTPUT	BCD	Output	<p>The first 30 input numbers plus the mean and standard deviation of the following:</p> <ul style="list-style-type: none"> (1) LRDP - LRDY (2) LRDY - previous LRDP (3) LRDP - previous LRDP (4) LDOC - previous LDUN (5) LDLK - LDOC (6) LIFT - LDLK (7) CPSN - LIFT (8) LAND - CPSN (9) LUND - LAND

MAIN WRTprt

SUBROUTINE ACCTY (1)

Purpose

To accumulate lighter type statistics

Calling Sequence

CALL ACCTY(SUMTY, SUMSQTY, NOTYF, NOTYR, N,SAME)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
SUMTY	Real	9,2	Input & Output	(1,1) Sum of forward LRDP-LRDY (2,1) Sum of forward LRDY-Pre- vious LRDP (3,1) Sum of forward LRDP-pre- vious LRDP (4,1) Sum of forward LDOC-pre- vious LDUN (5,1) Sum of forward LDLK-LDOC (6,1) Sum of forward LIFT-LDLK (7,1) Sum of forward CPSN-LIFT (8,1) Sum of forward LAND-CPSN (9,1) Sum of forward LUND-LAND (1,2) - (9,2) Sums of the same 9 times as above except retro- grade instead forward.
SUMSQTY	Real	6,2	Input & Output	Corresponding sums of squares of times above.
NOTYF	Integer	9	Input & Output	Number of each of the above forward times.
NOTYR	Integer	9	Input & Output	Number of each of the above retrograde times.
N	Integer	32	Input	(1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Direction (F/R) (9) LRDY (10) LRDP (11) LRDP - LRDY (12) LRDY - previous LRDP (13) LRDP - previous LRDP (14) Cargo type (15) Cargo ID (16) Operator ID

MAIN WRTprt

SUBROUTINE ACCTY (1) (CONTINUED)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
				(17) LDOC
				(18) LDOC - previous LDUN
				(19) LDLK
				(20) LDLK - LDOC
				(21) LIFT
				(22) LIFT - LDLK
				(23) CPSN
				(24) CPSN - LIFT
				(25) LAND
				(26) LAND - CPSN
				(27) LDUN
				(28) LDUN - LAND
				(29) Delay start
				(30) Delay type
				(31) Delay sequence number
				(32) Minutes elapsed from the beginning of the test.
SAME	Logical	-	Input	Equals false when lighter part of record is different from previous record and equals true when lighter part of record is the same as previous record.

Common Blocks

NONE

Subroutines Called

NONE

MAIN WTRPT

SUBROUTINE INTY (2)

Purpose

Initialize statistics parameters.

Calling Sequence

CALL INTY(SUMTY, SUMSQTY, NOTYF, NOTYR)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
SUMTY	Real	9,2	Output	(1,1) Sum of forward LRDP-LRDY (2,1) Sum of forward LRDY-pre- vious LRDP (3,1) Sum of forward LRDP-pre- vious LRDP (4,1) Sum of forward LDOC-pre- vious LDUN (5,1) Sum of forward LDLK-LDOC (6,1) Sum of forward LIFT-LDLK (7,1) Sum of forward CPSN-LIFT (8,1) Sum of forward LAND-CPSN (9,1) Sum of forward LUND-LAND (1,2) - (9,2) Sums of the same 9 times as above except retro- grade instead forward.
SUMSQTY	Real	9,2	Output	Corresponding sums of squares of times above.
NOTYF	Integer	9	Output	Number of each of the above forward times.
NOTYR	Integer	9	Output	Number of each of the above retrograde times.

Common Blocks

NONE

Subroutines Called

NONE

MAIN WTRPT

SUBROUTINE PGBRK (3)

Purpose

To write page headings and to print lighter/ argo records and statistics.

Calling Sequence

CALL PGBRK(FIRST, OFAL, ODATE, OSHIFT, N, KPGBR, OLTRTY, OFR, SUMTY, SUMSQTY, NOTYF, NOTYR, SAME)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
FIRST	Logical	-	Input	First call indicator
OFAC	Real	-	Output	Current Facility ID
ODATE	Real	-	Output	Current date
N	Real	-	Input	(1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Direction (F/R) (9) LRDY (10) LRDP (11) LRDP-LRDY (12) LRDY-previous LRDP (13) LRDP-previous LRDP (14) Cargo type (15) Cargo ID (16) Operator ID (17) LDOC (18) LDOC-previous LDUN (19) LDLK (20) LDLK-LDOC (21) LIFT (22) LIFT-LDLK (23) CPSN (24) CPSN-LIFT (25) LAND (26) LAND-CPSN (27) LDUN (28) LDUN-LAND (29) Delay start (30) Delay type (31) Delay sequence number (32) Minutes elapsed from the beginning of the test.

MAIN WRTRPT

SUBROUTINE PGBRK (3) (CONTINUED)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
KPGBR	Logical	-	Output	Page break indicator
OLTRTY	Real	-	Output	Current lighter type
OFR	Real	-	Output	Current forward/retrograde indicator.
SUMTY	Real	-	Input	(1,1) Sum of forward LRDP-LRDY (2,1) Sum of forward LRDY-pre- vious LRDP (3,1) Sum of forward LRDP-pre- vious LRDP (4,1) Sum of forward LDOC-pre- vious LDUN (5,1) Sum of forward LDLK-LDOC (6,1) Sum of forward LIFT-LDLK (7,1) Sum of forward CPSN-LIFT (8,1) Sum of forward LAND-CPSN (9,1) Sum of forward LUND-LAND (1,2) - (9,2) Sums of the same 9 times as above except retro- grade instead forward.
SUMSQTY	Real	-	Input	Corresponding sums of squares of times above.
NOTYR	Integer	3	Input	Number of each of the above retrograde times.
NOTYF	Integer	3	Input	Number of each of the above forward times.
SAME	Logical	-	Input	Same lighter part of record indicator.

Common Blocks

NONE

Subroutines Called

ACCTY
INTY
PRNIND
PRNTST

MAIN WRTRPT

SUBROUTINE PRNIND (4)

Purpose

To write individual lighter/cargo records.

Calling Sequence

CALL PRNIND (N, SAME)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
N	Integer	32	Input	(1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Direction (F/R) (9) LRDY (10) LRDP (11) LRDP-LRDY (12) LRDY-previous LRDP (13) LRDP-previous LRDP (14) Cargo type (15) Cargo ID (16) Operator ID (17) LDOC (18) LDOC-previous LDUN (19) LDLK (20) LDLK-LDOC (21) LIFT (22) LIFT-LDLK (23) CPSN (24) CPSN-LIFT (25) LAND (26) LAND-CPSN (27) LDUN (28) LDUN-LAND (29) Delay start (30) Delay type (31) Delay sequence number (32) Minutes elapsed from the beginning of the test.
SAME	Logical	-	Input	Same lighter part of record indicator.

MAIN WRTRPT

SUBROUTINE PRNIND (4) (CONTINUED)

Common Blocks

NONE

Subroutines Called

NONE

MAIN WRTRPT

SUBROUTINE PRNTST (5)

Purpose

To compute and print lighter type statistics.

Calling Sequence

CALL PRNTST (SUMTY, SUMSQTY, NOTYF, NOTYR)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
SUMTY	Real	9,2	Input	(1,1) Sum of forward LRDP-LRDY (2,1) Sum of forward LRDY-pre- vious LRDP (3,1) Sum of forward LRDP-pre- vious LRDP (4,1) Sum of forward LDOC-pre- vious LDUN (5,1) Sum of forward LDLK-LDOC (6,1) Sum of forward LIFT-LDLK (7,1) Sum of forward CPSN-LIFT (8,1) Sum of forward LAND-CPSN (9,1) Sum of forward LUND-LAND (1,2) - (9,2) Sums of the same 9 times as above except retro- grade instead forward. Corresponding sums of squares of times above.
SUMSQTY	Real	9,2	Input	
NOTYF	Integer	9	Input	Number of each of the above forward times.
NOTYR	Integer	9	Input	Number of each of the above retrograde times.

Common Blocks

NONE

Subroutines Called

NONE

MAIN WRTRPT

SUBROUTINE CHECK (6)

Purpose

To determine if the present lighter record is the same as the previous lighter record.

Calling Sequence

CALL CHECK(N, SAME)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
N	Integer	32	Input	(1) Facility ID (2) Julian date (3) Shift (4) Lighter type (5) Lighter ID (6) Lighter cycle (7) Lighter position (8) Direction (F/R) (9) LRDY (10) LRDP (11) LRDP-LRDY (12) LRDY-previous LRDP (13) LRDP-previous LRDP (14) Cargo type (15) Cargo ID (16) Operator ID (17) LDOC (18) LDOC-previous LDUN (19) LDLK (20) LDLK-LDOC (21) LIFT (22) LIFT-LDLK (23) CPSN (24) CPSN-LIFT (25) LAND (26) LAND-CPSN (27) LDUN (28) LDUN-LAND (29) Delay start (30) Delay type (31) Delay sequence number (32) Minutes elapsed from the beginning of the test.
SAME	Logical	-	Output	Equals false when lighter part of record is different from previous record and equals true when lighter part of record is the same as previous record.

MAIN WRTRPT

SUBROUTINE CHECK (6) (CONTINUED)

Common Blocks

NONE

Subroutines Called

NONE

AD-A131 715

JOINT LOGISTICS-OVER-THE-SHORE (LOTS) MAIN TEST
AUTOMATED DATA BASE REDUCTION PROGRAMS(U) ORI INC
SILVER SPRING MD H CASEY ET AL. MAR 79 ORI-TR-1477

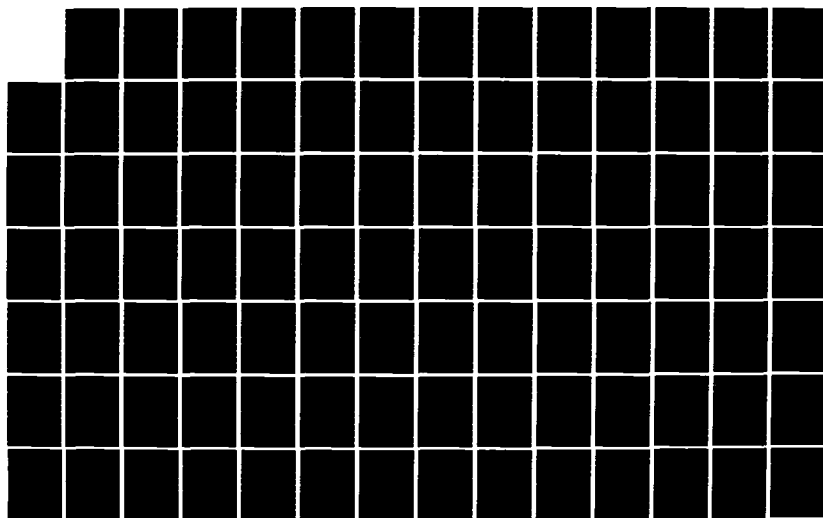
3/6

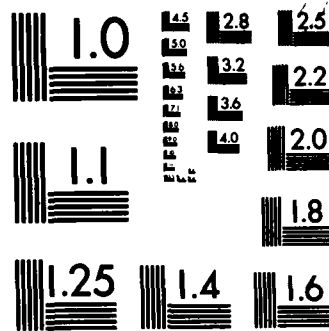
UNCLASSIFIED

MDA903-75-C-0016

F/G 9/2

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

V. CRANE CYCLE TIMES PROGRAM

INTRODUCTION

The purpose of the Crane Cycle Times Program is to select crane cycles that have certain attributes that are input to the program. The mean, standard deviation, median and as an option a histogram are computed for the selected cycles. The statistics are accumulated by shift and phase.

A sample output is given in Figure 5.1.

PROGRAM DESCRIPTION

The Crane Cycle Times Program is composed of a System 2000 retrieval program, two main programs and 24 routines.

A description of each program is given starting with the System 2000 retrieval program followed by the two main programs. The description of each main program contains a structure diagram, a flow chart of the main program followed by a write-up of each main program and routine.

SYSTEM 2000 RETRIEVALS FOR PROGRAM MAGIC

Purpose

To retrieve crane cycle data from the LOTS System 2000 Data Base and to write the data on a file.

<u>FILE NAME</u>	<u>TYPE</u>	<u>USE</u>	<u>DESCRIPTION</u>
RETRS2K	BCD	OUTPUT	(1) Facility ID (2) Julian date (3) Shift (4) Cargo ID (5) Cargo cycle (6) Discharging carrier type (7) Discharging carrier ID (8) Receiving carrier type (9) Receiving carrier ID (10) Sling/lifting device (11) Crane operator (12) Cargo event name (13) Cargo event time (LDOC or DELY)

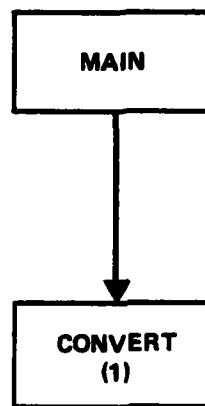


FIGURE 5.2. MAGIC STRUCTURE

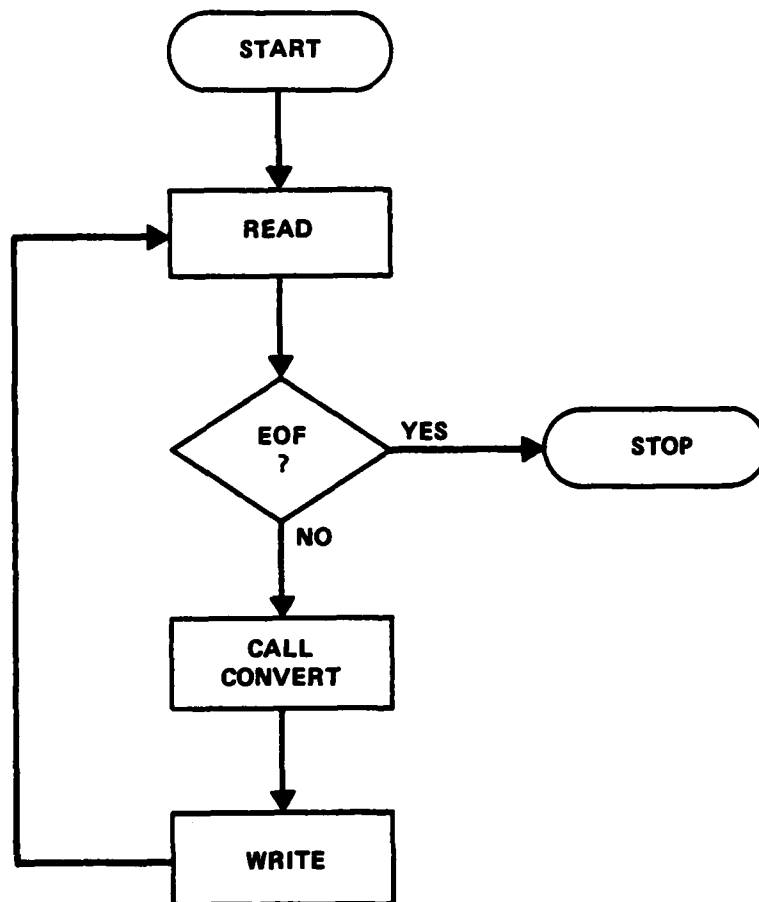


FIGURE 5.3. MAIN PROGRAM (MAGIC) FLOW DIAGRAM

MAIN PROGRAM MAGIC

Purpose

To convert Julian data, shift, and hours and minutes to minutes elapsed from the beginning of the test.

Program Statement

PROGRAM MAGIC (TAPE1, TAPE2, OUTPUT)

<u>FILE NAME</u>	<u>TYPE</u>	<u>USE</u>	<u>DESCRIPTION</u>
TAPE1	BCD	Input	(1) Facility ID (2) Julian date (3) Shift (4) Cargo ID (5) Cargo cycle (6) Discharging carrier type (7) Discharging carrier ID (8) Receiving carrier type (9) Receiving carrier ID (10) Sling/lifting device (11) Crane operator (12) Cargo event name (13) Cargo event time (LDOC or DELY)
TAPE2	BCD	Output	(1) Facility ID (2) Julian date (3) Shift (4) Cargo ID (5) Cargo cycle (6) Discharging carrier type (7) Discharging carrier ID (8) Receiving carrier type (9) Receiving carrier ID (10) Sling/lifting device (11) Crane operator (12) Cargo event name (13) Cargo event time (LDOC or DELY) (14) Elapsed time in minutes from the beginning of the test.
OUTPUT	BCD	Output	Contains system messages, if any

MAIN MAGIC

SUBROUTINE CONVERT 1

Purpose

Converts day, shift, hours, and minutes to minutes from the beginning of the test.

Calling Sequence

CALL CONVERT (NDAY, NSHIFT, NHM, MINS, MTIME)

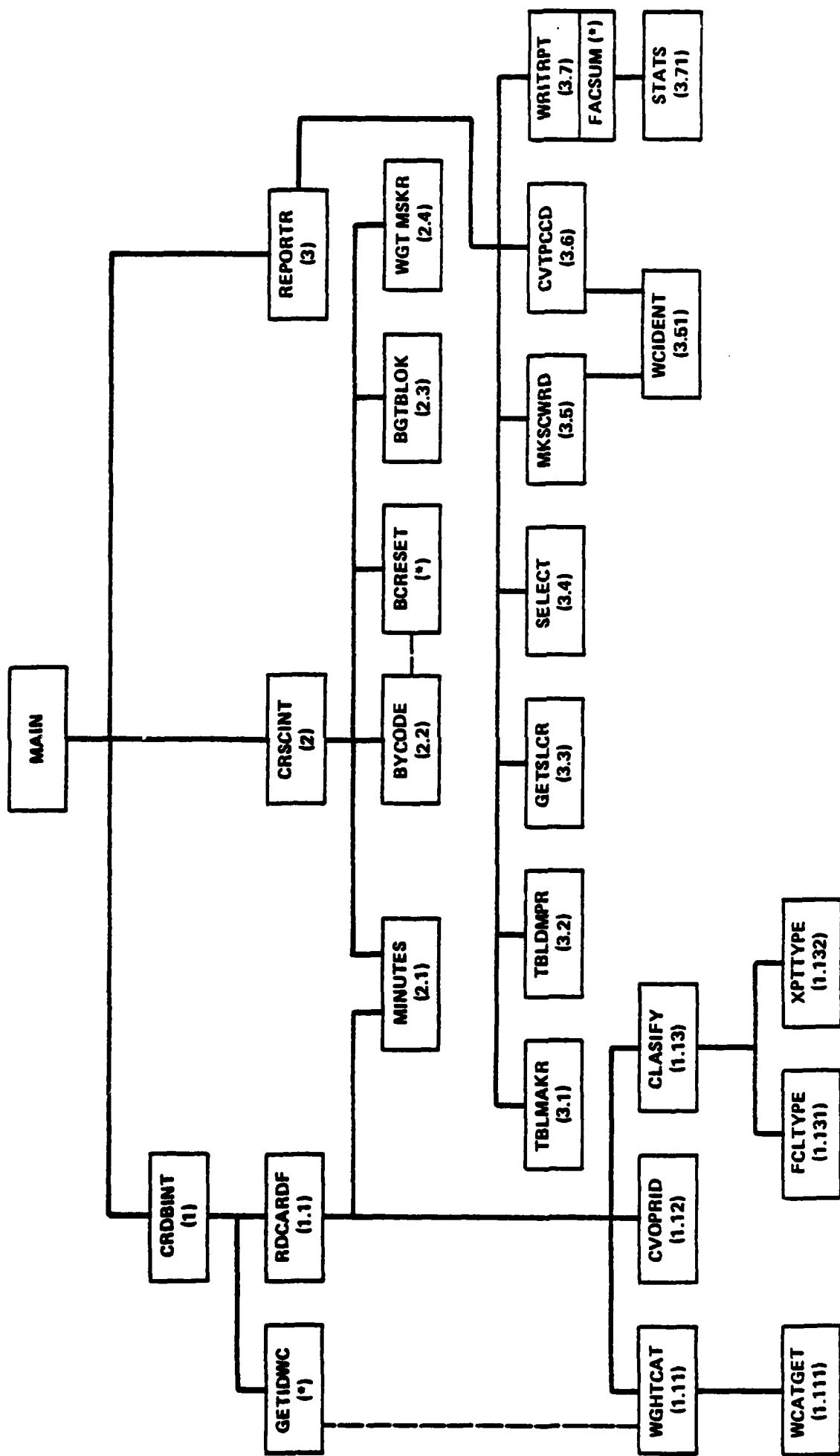
<u>Variable</u>	<u>Type</u>	<u>Dimension</u>	<u>Use</u>	<u>Description</u>
NDAY	Integer	-	Input	Julian date.
NSHIFT	Integer	-	Input	Shift 1 or shift 2.
NHM	Integer	-	Input	Time of day in hours and minutes (military time).
MINS	Integer	-	Output	Minutes elapsed from the beginning of the test.
NTIME	Integer	-	Output	Time of day in hours and minutes. Times after midnight on shift 2 have 2400 added to them

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
None	None	None

Subroutines Called

None



*ENTRY POINTS

FIGURE 5.4. CYCLOPS STRUCTURE

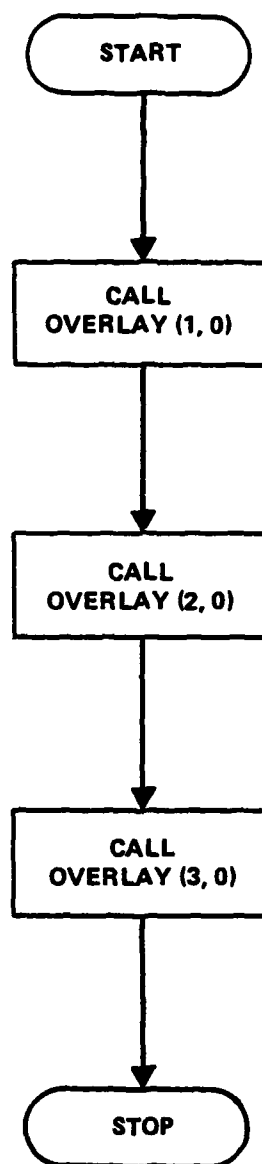


FIGURE 5.5. MAIN PROGRAM (CYCLOPS) FLOW DIAGRAM

MAIN PROGRAM CYCLOPS

Purpose

To calculate crane cycles and to produce a statistical report on selected cycles.

Program Statement

PROGRAM CYCLOPS (CARDF, SCFILE, DBINTR, SCINTR, REPORT, OUTPUT, CGIDTS)

<u>FILE NAME</u>	<u>TYPE</u>	<u>USE</u>	<u>DESCRIPTION</u>
CARDF	BCD	Input	(1) Facility ID (2) Julian date (3) Shift (4) Cargo ID (5) Cargo cycle (6) Discharging carrier type (7) Discharging carrier ID (8) Receiving carrier type (9) Receiving carrier ID (10) Sling/lifting device (11) Crane operator (12) Cargo event name (13) Cargo event time (14) Elapsed time in minutes from the beginning of the test.
SCFILE	BCD	Input	Selection Criteria Input File (1) Direction-of-movement (2) Cycle Interruption code (3) Sling/lifting device (4) Operator ID (5) Cycle type (6) Weight category (7) Lighter sequence (8) Special transporter (9) Cycle start lower limit (10) Cycle start upper limit (11) Cycle duration lower limit (12) Cycle duration upper limit (13) Number of histogram bins (14) Bin size (15) Print histogram (16) Title (17) Sequence number

<u>FILE NAME</u>	<u>TYPE</u>	<u>USE</u>	<u>DESCRIPTION</u>
DBINTR	BINARY	Input/ Output	(1) Direction of movement (2) Cycle type (3) Operator ID (4) Sling/lifting device (5) Weight category (6) Cycle start (7) Cycle duration (8) Special transporter (9) Lighter sequence (10) Cycle interruption c
SCINTR	BINARY	Output	This file contains the s parameters as the SCFILE file descr above. SCINTR is an expanded version of SCFILE.
REPORT	BCD	Output	This file contains the data of each SCINTR record followed by the selected cycles as given in the DBINTR file.
Output	BCD	Output	This file contains the SCINTR data and the statistical report.

PROGRAM CRDBINT (1)

Purpose

To create the data base intermediate file.

OVERLAY (1,0)

Routines Called

RDCARDF
GETIDWC

PROGRAM CRDBINT

FUNCTION RDCARDF (1.1)

Purpose

To read the "Cargo Activities Raw Data File"

Calling Sequence

RDCARDF (FP1, FP2, FP3, FP4, FP5, FP6, FP7, FP8, FP9, FP10, FP11, FP12, FP13)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
FP1	Integer	-	Output	Facility ID
FP2	Integer	-	Output	Julian date
FP3	Integer	-	Output	Shift
FP4	Integer	-	Output	Cargo ID
FP5	Integer	-	Output	Cargo cycle
FP6	Integer	-	Output	Discharging carrier type
FP7	Integer	-	Output	Discharging carrier ID
FP8	Integer	-	Output	Receiving carrier type
FP9	Integer	-	Output	Receiving carrier ID
FP10	Integer	-	Output	Sling/lifting device
FP11	Integer	-	Output	Crane operator
FP12	Integer	-	Output	Cargo event name
FP13	Integer	-	Output	Cargo event time

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
DBINFO		DBNAME, DBDATE, DBTIME
NCDREC		NCDREC

Routines Called

WGHTCAT
CVOPRID
CLASIFY
MINUTES

PROGRAM CRDBINT

FUNCTION WGHTCAT (1.11)

Purpose

To look up and return a weight code for a given cargo ID.

Calling Sequence

WGHTCAT (CGID)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
CGID	Integer	-	Input	Cargo ID

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
WGTCOM	,	WCCOUNT

Routines Called

WCATGET

PROGRAM CRDBINT

FUNCTION WCATGET (1.111)

Purpose

To determine the weight category of a container.

Calling Sequence

WCATGET (WEIGHT, LONGLTR)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
WEIGHT	Real	-	Input	Container weight
LONGLTR	Integer	-	Input	Equal 1 indicates 40 ft container.

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
WGTCOM		WCCOUNT

Routines Called

NONE

PROGRAM CRDBINT

FUNCTION CVOPRID (1.12)

Purpose

To convert operator ID from a 2-digit number to a letter code in the range A-L.

Calling Sequence

CVOPRID (OPERIDT)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
OPERIDT	Integer	-	Input	Operator ID

Common Blocks

NONE

Routines Called

NONE

PROGRAM CRDBINT

FUNCTION CLASIFY (1.13)

Purpose

To determine direction-of-movement, cycle type, and discharger/receiver break code for indicated discharger, receiver, facility ID combination.

Calling Sequence

CLASIFY (FACILID, DSCHXPT, RECVXPT, DIRMVMT, CYCTYPE, XPTBRK)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
FACILID	Integer	-	Input	Facility ID
DSCHXPT	Integer	-	Input	Discharging transporter
RECVXPT	Integer	-	Input	Receiving transporter
DIRMVMT	Integer	-	Output	Forward or Retrograde code
CYCTYPE	Integer	-	Output	Cycle type
XPTBRK	Integer	-	Output	Transporter break code

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
NCDREC	NCDREC	

Routines Called

FCLTYPE
XPTTYPE

PROGRAM CRDBINT

FUNCTION FCLTYPE (1.131)

Purpose

To determine facility type.

Calling Sequence

FCLTYPE (FCNAME, FT)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
FCNAME	Integer	-	Input	Facility name
FT	Integer	-	Output	Facility type (land or sea)

Common Blocks

NONE

Routines Called

NONE

PROGRAM CRDBINT

FUNCTION XPTTYPE (1.132)

Purpose

To determine type of transporter.

Calling Sequence

XPTTYPE (XPTNAME, XT)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
XPTNAME	Integer	-	Input	Transporter name
XT	Integer	-	Output	Transporter type (STOR, NDDC, LAND, LIGHTER)

Common Blocks

NONE

Routines Called

NONE

PROGRAM CRSCINT (2)

Purpose

To create selection criteria intermediate file.

OVERLAY (2,0)

Routines Called

MINUTES
BYCODE
BGTBLOK

PROGRAM CRSCINT

FUNCTION MINUTES (2.1)

Purpose

To convert hours/minutes code to minutes

Calling Sequence

MINUTES (HHMM)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
HHMM	Integer	-	Input	Hours/minutes code

Common Blocks

NONE

Routines Called

NONE

PROGRAM CRSCINT

FUNCTION BYCODE (2.2)

Purpose

To test for a "BY" code and set up BY-generation table.

Calling Sequence

BYCODE (SCFIELD, SCNUM, RETCODE)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
SCFIELD	Integer	-	Input	Selection criteria
SCNUM	Integer	-	Input	Number of selection criteria
RETCODE	Integer	-	Output	(1) =YES, set proper element of BY-generation table to SCNUM (2) =NO, user selected same BY number more than once (3) =ERR, not a valid BY code

Common Blocks

Block Name
BST

Input

Output
BYGENTB

Routines Called

NONE

PROGRAM CRSCINT

FUNCTION BGTBLOK (2.3)

Purpose

To see if BY-generation table is set up properly.

Calling Sequence

BGTTBLOK (DUMMY)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
DUMMY	Integer	-	Input	Dummy argument

Common Block

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
BGT	BYGENTB	

Routines Called

NONE

PROGRAM CRSCINT

FUNCTION WGTMSKR (2.4)

Purpose

To create a weight mask from individual weight category selection codes.

Calling Sequence

NGTMSKR (WGTCODE, RTNMASK)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
WGTMSKR	Integer	6	Input	Weight category selection code.
RTNMASK	Integer	-	Output	Weight Mask.

Common Block

None

Routines Called

None

PROGRAM RPTORTR (3)

Purpose

To supervise generation of cycle data statistics report.

OVERLAY (3,0)

Routines Called

TBLMAKR
TBLDMPR
GETSLCR
SELECT
MKSCWRD
CVTPCCD
WRITRPT

PROGRAM REPORTR

FUNCTION TBLMAKR (3.1)

Purpose

To construct facility table

Calling Sequence

TBLMAKR (FACNAME)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
FACNAME	Integer	-	Input	Facility Name

Common Blocks

Block Name

Input

Output

BLANK

DSILAST, OPDATE, OPSHFT,
FTBINDX, DIRMUMT, CYTYPE,
SGLFDEV, OPERID, WGHTCAT,
CYSTART, CYDURAT, SPCLXPT,
LGTRSEQ, CYINTER

Routines Called

NONE

PROGRAM REPORTR

SUBROUTINE TBLDMPR (3.2)

Purpose

To output Facility table.

Calling Sequence

TBLDMPR (FACNAME)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
FACNAME	Integer	-	Input	Facility name

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
BLANK	DSILAST, OPDATE, OPSHFT, FTBINDEX, DIRMVM, CYTYPE, SGLFDEV, OPERID, WGHTCAT, CYSTART, CYDURAT, SPCLXPT, LGTRSEQ, CYINTER	

Routines Called
NONE

PROGRAM REPORTR

FUNCTION GETSLCR (3.3)

Purpose

To read selection criteria from the intermediate file created by CRSCINT.

Calling Sequence

GETSLCR (SELCRIT, TITLE, BKTSIZE, NBUKET, PRTHIST)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
SELCRIT	Integer	12	Output	(1) Direction-of-movement (2) Cycle interruption (3) Sling/lifting device (4) Operator ID (5) Cycle type (6) Weight category (7) Lighter sequence (8) Special transporter (9) Cycle start lower limit (10) Cycle start upper limit (11) Cycle duration lower limit (12) Cycle duration upper limit
TITLE	Integer	4	Output	Card title
BKTSIZE	Integer	-	Output	Bin size for statistics report
NBUKET	Integer	-	Output	Number of buckets for statistics report
PRTHIST	Integer	-	Output	Equals yes to print histogram.

Common Blocks

Block Name

Input

Output

NSCREC

NSCREC

Routines Called

NONE

PROGRAM REPORTR

FUNCTION SELECT (3.4)

Purpose

To select and return a set of cycle times.

Calling Sequence

SELECT (SC, CYCLES, NCYCLES, SDATE, SSHIFT)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
SC	Integer	12	Input	Selection criteria array (see function GETSLCR)
CYCLES	Integer	200	Output	Selected cycle times
NCYCLES	Integer	-	Output	Number of cycles selected
SDATE	Integer	-	Output	Selected cycles date
SSHIFT	Integer	-	Output	Selected cycles shift

Common Blocks

Block Name

Input

Output

NSCREC
BLANK

NSCREC
DSILAST, OPDATE, OPSHIFT,
FTBINDX, DIRMVMT, CYTYPE,
SGLFTDEV, OPERID, WGHTCAT,
CYSTART, CYDURAT, SPCLXPT,
LGHTRSEQ, CYINTER

Routines Called

NONE

PROGRAM REPORTR

SUBROUTINE MKSCWRD (3.5)

Purpose

To fabricate selection criteria title words for subroutine WRITRPT.

Calling Sequence

CALL NKSCWRD (SCCODE, SCWORD)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
SCCODE	Integer	12	Input	Selection criteria code
SCWORD	Integer	10	Output	Selection criteria word

Common Blocks

NONE

Routines Called

WCIDENT

PROGRAM REPORTR

FUNCTION WCIDENT (3.51)

Purpose

To encode a 10-character word with a printable weight code designator, based on bits set in input parameter WTCODE.

Calling Sequence

WCIDENT (WTCODE)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
WTCODE	Integer	-	Input	Container weight code

Common Blocks

NONE

Routines Called

NONE

PROGRAM REPORTR

SUBROUTINE CVRPCCD (3.6)

Purpose

To convert cycles and to print selected cargo cycle data.

Calling Sequence

CALL CVTPCCD (RCTIMES, NCYRETR, TITLE, SCWORDS, BRKKEYS, PRTHIST, BRTSIZE, NBUCKET)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
RCTIMES	Integer	NCYRETR	Input	Cycle time indices
NCYRETR	Integer	-	Input	Number of cycle indices
TITLE	Integer	4	Input	Title from selection criteria record
SCWORDS	Integer	10	Input	Selection criteria words
BRKKEYS	Integer	3	Input	Facility, date, and shift break key
PRTHIST	Integer	-	Input	If yes, then print histogram
BKTSIZE	Integer	-	Input	Histogram bin size
NBUCKET	Integer	-	Input	Number of histogram bins

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
BLANK	DSILAST, OPCODE, OPSHIFT, FTBINDX, DIRMVMT, CYTYPE, SGLFDEV, OPERID, WGHTEAT, CYSTART, CYDURAT, SPCLXPT, LGTRSEQ, CYINTR	

Routines Called

WCIDENT

PROGRAM REPORTR

SUBROUTINE WRITRPT (3.7)

Purpose

To call subroutine STATS to compute statistics (mean, standard deviation, and optionally a median and histogram) based on a set of input cycle times and, when appropriate (i.e., at each facility break), shift-accumulated statistics. Then to print these out with appropriate headings. This routine is called each time there is a break on facility, date or shift. An entry point in this routine is used to print shift-accumulated statistics whenever facility changes.

Calling Sequence

CALL WRITRPT (CYTIM, NOTIMS, TITLE, SELCRI, BRKEYS, HISTSW, BINSIZ, NOBIN)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
CYTIM	Integer	200	Input	Selected cycle times
NOTIMS	Integer	-	Input	Number of cycle times
TITLE	Integer	4	Input	Title of run
SELCRI	Integer	10	Input	Selection criteria
BRKEYS	Integer	3	Input	(1) Facility ID (2) Julian date (3) Shift
HISTSW	Integer	-	Input	If yes, then print histogram
BINSIZ	Real	-	Input	Histogram bin size
NOBIN	Integer	-	Input	Number of histogram bins

Common Blocks

<u>Block Name</u>	<u>Input</u>	<u>Output</u>
SHFSTAT	SHSUM, SHSUMSQ, SHBINFR, SNOTIMS	

Routines Called

STATS

PROGRAM REPORTR

SUBROUTINE STATS (3.71)

Purpose

To compute mean, standard deviation, median and (optionally) histogram values based on a set of input cycle times. In addition, to accumulate shift associated statistics.

Calling Sequence

CALL STATS (CYTIM, NOTIMS, HISTSW, BINSIZ, NOBIN, XBAR, SDEV, MEDIAN, BINFREQ, BINREL, BINCUM, SHIFT)

<u>VARIABLE</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>USE</u>	<u>DESCRIPTION</u>
CYTIM	Integer	200	Input	Selected cycle times
NOTIMS	Integer	-	Input	Number of cycle times
HISTSW	Integer	4	Input	Title of run
BINSIZ	Real	-	Input	Histogram bin size
NOBIN	Integer	-	Input	Number of histogram bins
XBAR	Real	-	Output	Mean
SDEV	Real	-	Output	Standard deviation
MEDIAN	Real	-	Output	Median
BINFREQ	Real	100	Output	Number of cycles in each bin
BINREL	Real	100	Output	Relative histogram bin frequency.
BINCUM	Real	100	Output	Cululative histogram bin frequency.

Common Blocks

Block Name

Input

Output

SHFSTAT

SHSUM, SHSUMSQ, SHBINFR,
SNOTIMS

Routines Called

NONE

COMMON BLOCK BLANK

<u>NAME</u>	<u>TYPE</u>	<u>DIMENSION</u>	<u>DESCRIPTION</u>
DSILAST	Integer	-	Number of cycle times
OPDATE	Integer	15	Julian Date
OPSHIFT	Integer	15	Shift
FTBINDX	Integer	15	Facility table pointer
DIRMVM	Integer	700	Direction-of-movement
CYTYPE	Integer	700	Cycle type
SGLFDEV	Integer	700	Sling/lifting device type
OPERID	Integer	700	Operator ID
WGHTCAT	Integer	700	Container weight category
CYSTART	Integer	700	Cycle starting time
CYDURAT	Integer	700	Cycle duration
SPCLXPT	Integer	700	Special transporter type, if any
LGTRSEQ	Integer	700	Container sequence number
CYINTER	Integer	700	Cycle interruption code, if any

COMMON BLOCK BST

BYGENTB	Integer	6	The six selection criteria priority numbers
---------	---------	---	---

COMMON BLOCK NSCREC

NSCREC	Integer	-	Number of selection criteria records
--------	---------	---	--------------------------------------

COMMON BLOCK SHFSTAT

SHSUM	Real	2	Shift sum of cycles times
SHSUMSQ	Real	2	Shift sum of cycles times squared
SHBINFR	Real	(100,2)	Histogram bin frequency for the shift
SNOTIMS	Integer	(2)	Number of cycles for the shift

COMMON BLOCK WGTCOM

WCCOUNT	Integer	7	Number of containers in each of seven weight categories
---------	---------	---	---

APPENDIX A
CONSISTENCY CHECK
PROGRAM LISTINGS

```

LSTGT,CM60000,SPUU,T500,GST
TASK(TN=LOTS,TA=12345,OS=ORISPM,TR=TS,PI=GST)
COMMENT.***** CTL. STMTS. ON GENJ4MATCHR1DATA *****
COMMENT.***** RETRIEVALS FROM JLMT4A *****
ATTACH,S2K,S2K260,ID=SYS2000,MR=1.
REQUEST,J4CIBCA,*PF.
REQUEST,J4CDVLT,*PF.
REQUEST,J4CDVLC,*PF.
REQUEST,J4CRVLT,*PF.
REQUEST,J4CRVLC,*PF.
S2K,CR=77.
CATALOG,J4CIBCA,J4CIBCA03,ID=LOTS.
CATALOG,J4CDVLT,J4CDVLT03,ID=LOTS.
CATALOG,J4CDVLC,J4CDVLC03,ID=LOTS.
CATALOG,J4CRVLT,J4CRVLT03,ID=LOTS.
CATALOG,J4CRVLC,J4CRVLC03,ID=LOTS.
*EOR
USER,LOTS; SHARED DBN IS JLMT4A;
REPORT FILE IS J4CIBCA;
LI/TITLE L(4),L(4),L(4),L(4),L(4),L(4),L(4),L(4)/
C1305,C1325,C1320,C1335,C1330,C1010,C1110,C1120,
OB C1010,C1110,C1120,C1305 WH C1305 EXISTS;
REPORT FILE IS J4CDVLT;
LI/TITLE L(4),L(4),L(4),L(4),L(4),L(4)/
C1720,C1510,C1530,C1010,C1110,C1120,
OB C1010,C1110,C1120,C1720 WH C1720 EXISTS;
REPORT FILE IS J4CDVLC;
LI/TITLE L(4),L(4),L(4),L(4),L(4),L(4)/
C2120,C2010,C2030,C1010,C1110,C1120,
OB C1010,C1110,C1120,C2120 WH C2120 EXISTS;
REPORT FILE IS J4CRVLT;
LI/TITLE L(4),L(4),L(4),L(4),L(4),L(4)/
C1820,C1510,C1530,C1010,C1110,C1120,
OB C1010,C1110,C1120,C1820 WH C1820 EXISTS;
REPORT FILE IS J4CRVLC;
LI/TITLE L(4),L(4),L(4),L(4),L(4),L(4)/
C2220,C2010,C2030,C1010,C1110,C1120,
OB C1010,C1110,C1120,C2220 WH C2220 EXISTS;
EXIT;
*EOR
*EOF

```

L5TGT,CM65000,SPUU,T100.GST
TASK(TN=LOTS,TA=12345,OS=ORISPM,TR=TS,PI=GST)
ATTACH,MATCHR1,MATCHR1OBJECT,ID=LOTS,MR=1.
ATTACH,CIBCA,J4CIBCA03,ID=LOTS,MR=1.
ATTACH,CDVLT,J4CDVLT03,ID=LOTS,MR=1.
ATTACH,CDVLC,J4CDVLC03,ID=LOTS,MR=1.
ATTACH,CRVLT,J4CRVLT03,ID=LOTS,MR=1.
ATTACH,CRVLC,J4CRVLC03,ID=LOTS,MR=1.
ATTACH,CGIDTS,CARGOIDFILE,ID=LOTS,MR=1.
MATCHR1.

PROGRAM MATCHR1

```
+ (CIBCA, CDVLT, CDVLC, CRVLT, CRVLC, OUTPUT, CGIDTS=200,
+ TAPE1=CIBCA, TAPE2=CDVLT, TAPE3=CDVLC,
+ TAPE4=CRVLT, TAPE5=CRVLC, TAPE6=OUTPUT, TAPE7=CGIDTS)
```

```
*-----*
* N.B. THIS PROGRAM MUST BE COMPILED WITH AN "FTN" COMMAND WHICH *
* INCLUDES THE "Z" OPTION, SINCE SEVERAL CALLS ARE MADE TO *
* SUBROUTINE ENTRY POINTS WITHOUT ARGUMENTS BEING PASSED. *
* (CF. FORTRAN EXTENDED 4 MANUAL (CDC 84000009), P. I-11-9) *
*-----*
*
* G. S. TRUJILLO *
* APRIL, 1978 *
*****
```

IMPLICIT INTEGER (A-Z)

LOGICAL VLD CG ID, GET CGID

```
COMMON /TITLE/ CDATE, CTIME, DB NAME
COMMON /NDRRECS/ N DL RECS, N DC RECS, N RL RECS, N RC RECS
COMMON /DRTERRS/ DL UNREF, DC UNREF, RL UNREF, RC UNREF,
+ DL DUPL, DC DUPL, RL DUPL, RC DUPL,
+ DL INVAL, DC INVAL, RL INVAL, RC INVAL
COMMON /NSSCDUP/ NSSC DUP
COMMON /NORMAN1/ PRV FID, PRV DATE, PRV SHFT
COMMON /CGTUNIT/ NCGUNIT
COMMON /NREPORT/ NREPORT
+ , N PAGE
COMMON /TBL UNIT/ NCIBCA, NCDVLT, NCDVLC, NCRVLT, NCRVLC

DATA NCIBCA, NCDVLT, NCDVLC, NCRVLT, NCRVLC, NREPORT
+ / 1 , 2 , 3 , 4 , 5 , 6 /
DATA NRECORD /0/, PRV CGID /0/, N DC NMCH, N RC NMCH, N DUPL /3*0/
DATA NCGUNIT /7/
DATA TOT CG IV, TOT DC NM, TOT RC NM, TOT DUPS, TOT NDUP /5 * 0/
DATA PRV FID, PRV DATE, PRV SHFT /3 * 1H /
DATA N PAGE, NSSC DUP /2 * 0/
DATA DL UNREF, DL DUPL, DL INVAL,
+ DC UNREF, DC DUPL, DC INVAL,
+ RL UNREF, RL DUPL, RL INVAL,
+ RC UNREF, RC DUPL, RC INVAL /12 * 0/
```

```
*-----*
* INITIALIZATION *
*-----*
```

REWIND NCIBCA
REWIND NCDVLT

```

REWIND NCDVLT
REWIND NCDVLC
REWIND NCRVLT
REWIND NCRVLC
REWIND NCGUNIT

```

```

*                                READ DATA BASE NAME FROM HEADER RECORD
  READ(NCIBCA,1000) DB NAME
1000 FORMAT(A9/)
  IF (DB NAME .NE. 1H ) GO TO 50
  WRITE (NREPORT,1005)
1005 FORMAT(*OCIBCA FILE DOES NOT CONTAIN TITLE LINE AS RECORD NO. 1*)
  STOP 50

*                                GET CURRENT DATE AND TIME FROM SYSTEM
  50 CDATE = DATE(JDUMMY)
  CTIME = TIME(JDUMMY)

*                                READ CARGO IDENT FILE
  IF (.NOT. GET CGID(JDUMMY)) STOP 1

*-----
*                                READ A RECORD FROM CIBCA FILE
  100 READ(NCIBCA,1010)
    + CARGO ID, DIS CAR, DC TYPE, REC CAR,
    + RC TYPE, FACIL ID, ODATE, SHIFT
1010 FORMAT(3X,8(A4,3X))

  IF(EOF(NCIBCA)) 7777, 270

*-----
*                                TEST FOR START OF NEW GROUP
  270 IF (FACIL ID .EQ. PRV FID .AND. ODATE .EQ. PRV DATE
    + .AND. SHIFT .EQ. PRV SHFT) GO TO 300
*-----

***** NEW GROUP PROCESSING *****
  SEQ = 0
  PRV CGID = PRV DSCR = PRV RC = 1H

*                                OUTPUT CURRENT FILE SEG. ERROR STATS.
  IF (NRECORD .NE. 0)
    + WRITE(NREPORT, 1015) N DC NMCH, N RC NMCH, N DUPL, N CG INVL
1015  FORMAT(*-#90XI5* DISCHARGING CARRIER NON-MATCHES*/
    + 91XI5* RECEIVING CARRIER NON-MATCHES*/
    + 91XI5* DUPLICATE RECORDS*/
    + 91XI5* INVALID CARGO IDENTIFIERS*)

  IF (NSSC DUP .NE. 0) WRITE(NREPORT,1016) NSSC DUP
1016 FORMAT(91XI5* NSSC DUPLICATE CODES*)
  TOT NDUP = TOT NDUP + NSSC DUP

```

```

*                               ACCUMULATE ERROR TOTALS
TOT CG IV = TOT CG IV + N CG INVL
TOT DC NM = TOT DC NM + N DC NMCH
TOT RC NM = TOT RC NM + N RC NMCH
TOT DUPL = TOT DUPL + N DUPL

*                               RESET ERROR COUNTERS
N DC NMCH = N RC NMCH = N DUPL = NSSC DUP = N CG INVL = 0

*                               DUMP DISCHARGE/RECEIVE CARRIER TABLES
IF (NRECORD .NE. 0) CALL TBL DUMP

*                               OUTPUT PAGE HEADING
NPAGE = NPAGE + 1
LINECNT = 0
WRITE(NREPORT,1006) DB NAME, FACIL ID, ODATE, SHIFT, CDATE,
+      CTIME, NPAGE
1006 FORMAT(*1*8X*DATA BASE *A9,8X*FACILITY *A4,8X*DATE *
+      A4,8X*SHIFT*A4,5X2(5XA9)5X*PAGE *I3//
+      4X*SEQ*5X*DISCHARGER*5X*RECEIVER*6X*CARGO*9X*DSCH CAR*
+      7X*D/C TYPE*7X*RECV CAR*7X*R/C TYPE*/)

*                               SAVE CURRENT BLOCK INFORMATION
*                               TO DETECT DATA SEGMENT BREAK
PRV FID = FACIL ID
PRV DATE = ODATE
PRV SHFT = SHIFT

*                               GET DISCHARGING/RECEIVING
*                               CARRIER DATA
CALL GDRDATA

*                               CLEAR 'NSSC CHK''S COMPARISON TABLE
CALL NSSC CLR

*                               RESET 'F DISC ID''S SCAN POINTERS
CALL FDI RSET

***** END OF NEW GROUP PROCESSING *****
***** PROCESS A RECORD FROM CIBCA FILE *****

300 NRECORD = NRECORD + 1
SEQ = SEQ + 1

*                               CHECK FOR DUPLICATE RECORD
IF (CARGO ID .NE. PRV CGID .OR. DIS CAR .NE. PRV DSCR
+      .OR. REC CAR .NE. PRV RC) GO TO 310
MSG1 = 10H*DUPLICATE
MSG2 = 10HRECORD ***
N DUPL = N DUPL + 1
GO TO 380

```

```

*                               SAVE CURRENT RECORD INFORMATION
*                               TO ALLOW DUPLICATE RECORD DETECTION
310 PRV CGID = CARGO ID
    PRV DSCR = DIS CAR
    PRV RC  = REC CAR
*
*                               MAKE SURE CARGO ID IS VALID
    IF (VLD CG ID(CARGO ID)) GO TO 305
    MSG1 = 10H**INVALID
    MSG2 = 10HCARGO ID**
    N CG INVL = N CG INVL + 1
    GO TO 380
*
*                               CHECK FOR STORAGE CODE
*                               AS DISCHARGING CARRIER TYPE
305 IF (DC TYPE .NE. 4HSTOR) GO TO 320
    MSG1 = 7HSTORAGE
    GO TO 330
320 IF (DC TYPE .NE. 4HNSSC) GO TO 325
    MSG1 = NSSC CHK(DIS CAR, SEQ)
    GO TO 330
*
*                               FIND DISCHARGING CARRIER RECORD
325 MSG1 = F DISC ID(CARGO ID, DIS CAR, SEQ)
    IF (MSG1 .EQ. 10H*NO MATCH*) N DC NMCH = N DC NMCH + 1
*
*                               CHECK FOR STORAGE CODE
*                               AS RECEIVING CARRIER TYPE
330 IF (RC TYPE .NE. 4HSTOR) GO TO 340
    MSG2 = 7HSTORAGE
    GO TO 380
340 IF (RC TYPE .NE. 4HNSSC) GO TO 350
    MSG2 = NSSC CHK(REC CAR, SEQ)
    GO TO 380
*
*                               FIND RECEIVING CARRIER RECORD
350 MSG2 = F RECV ID(CARGO ID, REC CAR, SEQ)
    IF (MSG2 .EQ. 10H*NO MATCH*) N RC NMCH = N RC NMCH + 1
*
*                               PRINT NEXT LINE OF CIBCA REPORT,
*                               AND NEW PAGE HEADING, IF REQUIRED
380 IF (LINECNT .LE. 52) GO TO 400
    LINECNT = 0
    NPAGE = NPAGE + 1
    WRITE(NREPORT,1006) DB NAME, FACIL ID, ODATE, SHIFT,
+      CDATE, CTIME, NPAGE
400 WRITE(NREPORT,1100)
+   SEQ, MSG1, MSG2, CARGO ID, DIS CAR, DC TYPE, REC CAR, RC TYPE
    LINECNT = LINECNT + 1
1100 FORMAT(1X,I6,7(5X,A10))
    GO TO 100

```

***** END OF CURRENT RECORD PROCESSING *****

```

*-----
*                               EOF ENCOUNTERED ON FILE CIBCA--
*                               PRINT RUN STATISTICS
*-----

```

7777 CONTINUE

```

WRITE(NREPORT,1015) N DC NMCH, N RC NMCH, N DUPL, N CG INVL
CALL TBL DUMP
WRITE(NREPORT,1177) DBNAME, CTIME, CDATE
WRITE(NREPORT,1190) NRECORD, TOT DC NM, TOT RC NM,
+               TOT DUPS, TOT CG IV, TOT NDUP

WRITE(NREPORT,1191) N DL RECS
WRITE(NREPORT,1200) DL UNREF, DL DUPL, DL INVAL,
+               DL UNREF + DL DUPL + DL INVAL

WRITE(NREPORT,1192) N DC RECS
WRITE(NREPORT,1200) DC UNREF, DC DUPL, DC INVAL,
+               DC UNREF + DC DUPL + DC INVAL

WRITE(NREPORT,1193) N RL RECS
WRITE(NREPORT,1200) RL UNREF, RL DUPL, RL INVAL,
+               RL UNREF + RL DUPL + RL INVAL

WRITE(NREPORT,1194) N RC RECS
WRITE(NREPORT,1200) RC UNREF, RC DUPL, RC INVAL,
+               RC UNREF + RC DUPL + RC INVAL

WRITE(NREPORT,1199) DL UNREF + DC UNREF + RL UNREF + RC UNREF,
+               DL DUPL + DC DUPL + RL DUPL + RC DUPL,
+               DL INVAL + DC INVAL + RL INVAL + RC INVAL

```

```

*-----
1177 FORMAT(*1*30X*ERROR SUMMARY FOR DATA BASE *A9* AT *
+               A9* ON *A10//)

```

1190 FORMAT(*0*

```

+       4XI5* TOTAL COUNT OF RECORDS IN D/R FILE*/
+       5XI5* TOTAL FILE DISCHARGING CARRIER NON-MATCHES*/
+       5XI5* TOTAL FILE RECEIVING CARRIER NON-MATCHES*/
+       5XI5* TOTAL FILE DUPLICATE RECORDS*/
+       5XI5* TOTAL INVALID CARGO IDENTIFIERS*/
+       5XI5* TOTAL NSSC CODE DUPLICATIONS*//)

```

```

1191 FORMAT(*0DISCHARGING LIGHTER*//5XI5* TOTAL RECORDS*)
1192 FORMAT(*0DISCHARGING LAND CARRIER*//5XI5* TOTAL RECORDS*)
1193 FORMAT(*0RECEIVING LIGHTER*//5XI5* TOTAL RECORDS*)
1194 FORMAT(*0RECEIVING LAND CARRIER*//5XI5* TOTAL RECORDS*)

```

```
1199 FORMAT(5X15* TOTAL NON-MATCHES*/  
+          5X15* TOTAL DUPLICATE RECORDS*/  
+          5X15* TOTAL INVALID CARGO IDENTIFIERS*)  
  
1200 FORMAT(5X15* NON-MATCHES*/5X15* DUPLICATE RECORDS*/  
+          5X15* INVALID CARGO IDENTIFIERS*/  
+          5X15* TOTAL ERRORS*//)  
END
```

SUBROUTINE GDRDATA

 * GET DISCHARGING/RECEIVING TRANSPORTER DATA FROM APPROPRIATE FILES.*

IMPLICIT INTEGER (A-Z)

COMMON /TBL UNIT/ NCIBCA, NCDVLT, NCDVLC, NCRVLT, NCRVLC
 LOGICAL FC FLAG

COMMON /NDRRECS/ N DL RECS, N DC RECS, N RL RECS, N RC RECS

COMMON /CDVLT/ CDLCID(300), CDLTID(300), CDLTYP(300), CDLSEQ(300),
 + NUMCDL
 COMMON /CDVLC/ CDCCID(300), CDCTID(300), CDCTYP(300), CDCSEQ(300),
 + NUMCDC
 COMMON /CRVLT/ CRLCID(300), CRLTID(300), CRLTYP(300), CRLSEQ(300),
 + NUMCRL
 COMMON /CRVLC/ CRCCID(300), CRCTID(300), CRCTYP(300), CRCSEQ(300),
 + NUMCRC
 DATA FC FLAG /.TRUE./

*-----
 * CARGO TRANS- XPORTER MATCHED INPUT FILE 1ST CALL
 * IDENT PORTER TYPE CODE TB UNIT NO. NAME FLAG

NUMCDL =
 + READTBL(CDLCID, CDLTID, CDLTYP, CDLSEQ, NCDVLT, 5HCDVLT, FC FLAG)
 N DL RECS = N DL RECS + NUMCDL

NUMCDC =
 + READTBL(CDCCID, CDCTID, CDCTYP, CDCSEQ, NCDVLC, 5HCDVLC, FC FLAG)
 N DC RECS = N DC RECS + NUMCDC

NUMCRL =
 + READTBL(CRLCID, CRLTID, CRLTYP, CRLSEQ, NCRVLT, 5HCRVLT, FC FLAG)
 N RL RECS = N RL RECS + NUMCRL

NUMCRC =
 + READTBL(CRCCID, CRCTID, CRCTYP, CRCSEQ, NCRVLC, 5HCRVLC, FC FLAG)
 N RC RECS = N RC RECS + NUMCRC

FC FLAG = .FALSE.

END

```

      INTEGER FUNCTION READTBL
      +      (CARG TBL, CARR TBL, CRTP TBL, SEQ TBL, NUNIT, FNAME, CALL1)
      *      READ ONE D/R FILE INTO A TABLE

*****
*      INPUT VALUES INTO THE CARGO, CARRIER, IDENTIFIER, AND CARRIER-TYPE*
*      ARRAYS FROM THE FILE WHOSE UNIT NUMBER IS PASSED AS THE VALUE OF *
*      FORMAL PARAMETER 'NUNIT', CLEARING AS MANY ELEMENTS OF THE 'SEQTBL' *
*      ARRAY. IF THE VALUE OF THE LOGICAL PARAMETER 'CALL1' IS .TRUE. *
*      (WHICH IT SHOULD BE ONLY THE FIRST TIME 'READ TBL' IS CALLED TO *
*      BUILD A PARTICULAR 'TABLE' (DEFINED AS THE COLLECTION OF ARRAYS *
*      CONTAINING DATA FOR A PARTICULAR CLASS OF CARGO CARRIER)), READ AND *
*      VALIDATE THE HEADER RECORD OF THE PARTICULAR FILE BEING READ. *
*
*      READ FROM THE INDICATED FILE ONLY WHILE THE VALUES OF THE FACILITY*
*      IDENTIFIER, DATE, AND SHIFT FIELDS OF THE RECORDS MATCH THE *
*      CORRESPONDING FIELDS OF THE 'CIRCA' FILE BEING READ BY THE MAIN *
*      PROGRAM, AS INDICATED BY THE CONTENTS OF THE VARIABLES CONTAINED. *
*      IN NAMED COMMON BLOCK 'NORMAN1' (CREDIT BEING EXTENDED TO *
*      N. SHUSTERMAN, ESQ., WHOSE IDEA IT WAS TO PASS THESE VALUES *
*      THROUGH COMMON, INSTEAD OF DOING SO THROUGH THE USE OF THE *
*      ROUTINE'S ARGUMENT LIST.) *
*****

```

IMPLICIT INTEGER (A-Z)

```

      INTEGER CARG TBL(300), CARR TBL(300), SEQ TBL(300), CRTP TBL(300)
      LOGICAL CALL1, VLD CG ID
      COMMON /NREPORT/ NREPORT
      COMMON /NORMAN1/ CUR FID, CUR DATE, CUR SHFT

```

```

*-----
*      VERIFY FILE HEADER ON FIRST CALL
      IF (.NOT. CALL1) GO TO 100
      READ(NUNIT, 1000) HFNAME
1000  FORMAT(A5/)

      IF (EOF(NUNIT)) 50, 60
      50  WRITE(NREPORT,1001) NUNIT
1001  FORMAT(*- *20(1H*)* NULL FILE ON INPUT UNIT *I2)
      STOP 2

      60  CONTINUE
*      IF (HFNAME .EQ. FNAME) GO TO 100
*      WRITE(NREPORT,1002) NUNIT, FNAME, HFNAME
*1002  FORMAT(*- *20(1H*)* WRONG FILE ON UNIT NO. *I2
*      +      * EXPECTING FILE *A5* -- READ FILE *A5)
*      STOP 211
*-----

```



```

100 READTBL = 0
   PRV CGID = PRV CARR = 0
   DO 300 I = 1, 300
       SEQ TBL(I) = 0
       READ(NUNIT,1010)
+       CARG TBL(I), CARR TBL(I), CRTF TBL(I),
+       FACIL ID, DATE, SHIFT
1010  FORMAT(3X,6(A4,3X))
       IF (EOF(NUNIT)) 777, 200

*           DETECT GROUP SEGMENT BREAK
200  IF((FACIL ID .NE. CUR FID) .OR. (DATE .NE. CUR DATE)
+      .OR. (SHIFT .NE. CUR SHFT)) GO TO 500

*           CHECK VALIDITY OF CARGO IDENTIFIER
       IF (.NOT. VLD CG ID(CARG TBL(I))) SEQ TBL(I) = -2

*           DETECT DUPLICATE RECORD
       IF (CARG TBL(I) .EQ. PRV CGID .A. CARR TBL(I) .EQ. PRV CARR)
+           SEQ TBL(I) = -1
       PRV CGID = CARG TBL(I)
       PRV CARR = CARR TBL(I)

300  CONTINUE

       WRITE(NREPORT,1050)NUNIT
1050  FORMAT(*- *20(1H*)* ARRAY OVERFLOW--READING FROM UNIT *I2)
       STOP 3

500  BACKSPACE NUNIT
777  READTBL = I - 1
      END

```

```

      INTEGER FUNCTION F DISC ID(CGID, CRID, CIBCASQ)
*****
*   CALL SCANTAB TO ATTEMPT TO LOCATE AN ENTRY IN EITHER OF THE   *
*   DISCHARGING TRANSPORTER TABLES WHOSE CARGO AND CARRIER IDENT *
*   MATCH THOSE PASSED BY THE CALLER.                               *
*****
      IMPLICIT INTEGER (A-Z)
      COMMON /CDVLT/ CDLCID(300), CDLTID(300), CDLTYP(300), CDLSEQ(300),
+              NUMCDL
      COMMON /CDVLC/ CDCCID(300), CDCTID(300), CDCTYP(300), CDCSEQ(300),
+              NUMCDC
      COMMON /CRVLT/ CRLCID(300), CRLTID(300), CRLTYP(300), CRLSEQ(300),
+              NUMCRL
      COMMON /CRVLC/ CRCCID(300), CRCTID(300), CRCTYP(300), CRCSEQ(300),
+              NUMCRC
      DATA LTRNAME, LCNAME / 5H LGHTR, 5H LANDC /
      DATA NOMATCH / 10H*NO MATCH*/
      DATA   CDL NXEL, CDC NXEL, CRL NXEL, CRC NXEL   / 4 * 1 /

*-----
200 F DISC ID = SCANTAB(CDLCID, CDLTID, CDLSEQ, NUMCDL,
+                    LTRNAME, CGID, CRID, CIBCASQ, CDL NXEL)
      IF (F DISC ID .NE. 0) RETURN

      F DISC ID = SCANTAB(CDCCID, CDCTID, CDCSEQ, NUMCDC,
+                    LCNAME, CGID, CRID, CIBCASQ, CDC NXEL)
      IF (F DISC ID .NE. 0) RETURN

      F DISC ID = NO MATCH
      RETURN

*-----
      ENTRY F RECV ID
*****
*   CALL SCANTAB TO ATTEMPT TO LOCATE AN ENTRY IN EITHER OF THE   *
*   RECEIVING TRANSPORTER TABLES WHOSE CARGO AND CARRIER IDENTS *
*   MATCH THOSE PASSED BY THE CALLER.                               *
*****
      400 F DISC ID = SCANTAB(CRLCID, CRLTID, CRLSEQ, NUMCRL,
+                    LTRNAME, CGID, CRID, CIBCASQ, CRL NXEL)
      IF (F DISC ID .NE. 0) RETURN
      F DISC ID = SCANTAB(CRCCID, CRCTID, CRCSEQ, NUMCRC,
+                    LCNAME, CGID, CRID, CIBCASQ, CRC NXEL)
      IF (F DISC ID .NE. 0) RETURN
      F DISC ID = NO MATCH
      RETURN

*-----
      ENTRY FDI RSET
*****
*   RESET "START SCAN" NSSC TABLE POINTERS.                       *
*****
      CDL NXEL = CDC NXEL = CRL NXEL = CRC NXEL = 1
      END

```

```

      INTEGER FUNCTION SCANTAB
      +(CARGTAB, CARRTAB, SQCIBCA, TABLIM, TABNAME, CARGID, CARRID,
      + NSEQ, STRTSCN)

```

```

*****
*      SCAN APPROPRIATE TRANSPORTER TABLE FOR A MATCH WITH THE
*      RECORD CURRENTLY BEING PROCESSED.  UPON RETURN, FORMAL PARAMETER
*      "STRT SCN" WILL BE SET TO THE FIRST ENTRY IN THE GROUP CURRENTLY
*      BEING SCANNED, WHERE "GROUP" IS DEFINED AS THOSE ENTRIES WITH THE
*      SAME CARGO ID.
*
*      RETURN A DISPLAY-CODED WORD INDICATING THE LOCATION WITHIN
*      ONE OF THE TABLES OF THE SEARCHED-FOR RECORD AS THE FUNCTION
*      VALUE IF THE SCAN IS SUCCESSFUL.
*****

```

```

      IMPLICIT INTEGER (A-Z)

```

```

      DIMENSION CARGTAB(TABLIM), CARRTAB(TABLIM), SQCIBCA(TABLIM)
      SHIFT1R(IDENT) = IDENT/2 .AND. 37777 77777 77777 77777 B

```

```

*-----

```

```

      IF (CARRID .NE. 4HSTOR) GO TO 400
      SCANTAB = 7HSTORAGE
      RETURN

```

```

400 SCANTAB = 0
      IF (TABLIM .EQ. 0 .OR. STRTSCN .GT. TABLIM
      + .OR. CARGID .EQ. 1HU) RETURN
      NEW STSC = STRT SCN
      CUR CGTB = CARGTAB(STRT SCN)
      DO 500 I = STRTSCN, TABLIM

```

```

*      TEST FOR ENTRY ALREADY HAVING MATCHED
*      OR BEING INELIGIBLE DUE TO AN
*      ERROR IN THE ENTRY (CF. "READTBL")

```

```

      IF (SQCIBCA(I) .NE. 0) GO TO 450
      IF (CARGID .NE. CARGTAB(I)) GO TO 450
      IF (CARRID .EQ. CARRTAB(I)) GO TO 700

```

```

*      CHANGE POINTER TO START OF SEQUENCE
*      OF ENTRIES WITH SAME CARGO ID
*      WHEN A BREAK IS FOUND

```

```

      IF (CARGTAB(I) .EQ. CUR CGTB) GO TO 450
      CUR CGTB = CARGTAB(I)
      NEW STSC = I

```

```

450      IF (SHIFT1R(CARGID) .LT. SHIFT1R(CARGTAB(I))) GO TO 800
500      CONTINUE
      GO TO 800

```

```

*          SAVE SEQ. NO. OF CIBCA RECORD WITHIN TRANSPORTER TABLE
700 SQCIBCA(I) = NSEQ
*          RETURN CODE OF FORM '<XNAME> NNNN'
      ENCODE(10,1000,SCANTAB) TABNAME, I
1000 FORMAT(A5,1X,I4)
800 STRTSCN = NEW STSC

      END

```

```

      FUNCTION NSSC CHK(IDENT, SEQ)
*****
*   ATTEMPT TO FIND IDENT IN TABLE OF SUPPOSEDLY-UNIQUE NSSC IDENTs   *
*****
      COMMON /NREPORT/ NREPORT
      COMMON /NSSCDUP/ NSSC DUP
      INTEGER IDNT TBL(200), LSTUSED, SEQ, EXTRACT
      EXTRACT(JWORD) = (JWORD .AND. 77777 77700 00000 00000B)
+                               .OR. 00000 00055 55555 55555B
*-----

      NSSC CHK = 4HNSSC
      IF (LSTUSED .EQ. 0) GO TO 550

      DO 500 I = 1, LSTUSED
        ID FR TAB = IDNT TBL(I)
        IF (IDENT .EQ. EXTRACT(ID FR TAB)) GO TO 700
500    CONTINUE

      550 LSTUSED = LSTUSED + 1
        IF (LSTUSED .LE. 200) GO TO 600
        WRITE(NREPORT,1010)
1010    FORMAT(*0*10X20(*-*)* OVERFLOW IN "NSSC CHK" IDENTIFIER TABLE*)
        STOP 4

      600 IDNT TBL(LSTUSED) =
+      (IDENT .AND. 77777 77700 00000 00000B) .OR. SEQ
      NSSC CHK = 4HNSSC
      RETURN

      700 ENCODE(10,1000,NSSCCHK) (ID FR TAB .AND. 7777B)
1000  FORMAT(*NSSCDUP*I3)
      NERR = NERR + 1
      NSSC DUP = NSSC DUP + 1
      RETURN

*-----
      ENTRY NSSC CLR
*****
*   RESET COUNT OF NO. OF TABLE ELEMENTS USED FOR PROCESSING   *
*   OF NEW DISCHARGING/RECEIVING TRANSPORTER GROUP.             *
*****
      LSTUSED = 0

      END

```

```

SUBROUTINE TBL DUMP
*
* GET ALL FOUR TRANSPORTER TABLES DUMPED
IMPLICIT INTEGER (A-Z)

DIMENSION NUMERRS(3)

COMMON /DRTERRS/ DL UNREF, DC UNREF, RL UNREF, RC UNREF,
+               DL DUPL, DC DUPL, RL DUPL, RC DUPL,
+               DL INVAL, DC INVAL, RL INVAL, RC INVAL

COMMON /CDVLT/ CDLCID(300), CDLTID(300), CDLTYP(300), CDLSEQ(300),
+             NUMCDL
COMMON /CDVLC/ CDCCID(300), CDCTID(300), CDCTYP(300), CDCSEQ(300),
+             NUMCDC
COMMON /CRVLT/ CRLCID(300), CRLTID(300), CRLTYP(300), CRLSEQ(300),
+             NUMCRL
COMMON /CRVLC/ CRCCID(300), CRCTID(300), CRCTYP(300), CRCSEQ(300),
+             NUMCRC
COMMON /PAGENUM/ N PAGE

*-----
IF (NUMCDL + NUMCDC + NUMCRL + NUMCRC .EQ. 0) RETURN

* PREPARE DUMPSTER FOR A NEW REPORT
CALL DPRESET

* DISCHARGING LIGHTER TABLE
100 IF (NUMCDL .EQ. 0) GO TO 200
CALL DMPSTER(CDLCID, CDLTID, CDLTYP, CDLSEQ, NUMCDL,
+ 27H DISCHARGING LIGHTER TABLE, NUMERRS)
DL UNREF = DL UNREF + NUMERRS(1)
DL DUPL = DL DUPL + NUMERRS(2)
DL INVAL = DL INVAL + NUMERRS(3)

* DISCHARGING LAND CARRIER TABLE
200 IF (NUMCDC .EQ. 0) GO TO 300
CALL DMPSTER(CDCCID, CDCTID, CDCTYP, CDCSEQ, NUMCDC,
+ 30H DISCHARGING LAND CARRIER TABLE, NUMERRS)
DC UNREF = DC UNREF + NUMERRS(1)
DC DUPL = DC DUPL + NUMERRS(2)
DC INVAL = DC INVAL + NUMERRS(3)

* RECEIVING LIGHTER TABLE
300 IF (NUMCRL .EQ. 0) GO TO 400
CALL DMPSTER(CRLCID, CRLTID, CRLTYP, CRLSEQ, NUMCRL,
+ 26H RECEIVING LIGHTER TABLE, NUMERRS)
RL UNREF = RL UNREF + NUMERRS(1)
RL DUPL = RL DUPL + NUMERRS(2)
RL INVAL = RL INVAL + NUMERRS(3)

```

```

*                                     RECEIVING LAND CARRIER TABLE
400 IF (NUMCRC .EQ. 0) RETURN
    CALL DMPSTER(CRCCID, CRCTID, CRCTYP, CRCSEQ, NUMCRC,
+       29H RECEIVING LAND CARRIER TABLE, NUMERRS)
    RC UNREF = RC UNREF + NUMERRS(1)
    RC DUPL  = RC DUPL  + NUMERRS(2)
    RC INVAL = RC INVAL + NUMERRS(3)
END

```

```

SUBROUTINE DMPSTER
+   (CARG TBL,CARR TBL,TYPE TBL,SEQ TBL,TBL SIZE,HEADING,NER ARAY)

*****
*   DUMP ONE OF THE FOUR TRANSPORTER TABLES.
*****

INTEGER CARG TBL(TBL SIZE), CARR TBL(TBL SIZE), NER ARAY(3),
+   SEQ TBL(TBL SIZE), TYPE TBL(TBL SIZE), TBL SIZE
DIMENSION HEADING(3)
COMMON /NORMAN1/ FACIL ID, DATE, SHIFT
COMMON /TITLE/ CDATE, CTIME, DB NAME
COMMON /NREPORT/ NREPORT, NPAGE

*-----

IF (TBL SIZE .EQ. 0) STOP 5

*           FORCE NEW PAGE IF NEW TABLE WILL
*           NOT FIT ENTIRELY ON CURRENT PAGE
IF (N LINES .EQ. 0 .OR. N LINES + TBL SIZE + 4 .GT. 47) N LINES = 48

N UNREF = N DUPL = N INVAL = 0
DO 50 I = 1, TBL SIZE
  IF (N LINES .LE. 47) GO TO 25
  N LINES = 0
  NPAGE = NPAGE + 1
  WRITE(NREPORT,1030)
+   DB NAME, FACIL ID, DATE, SHIFT, CDATE, CTIME, NPAGE
1030  FORMAT(*1*8X*DATA BASE *A9,8X*FACILITY *A4,8X*DATE *
+   A4,8X*SHIFT*A4,5X2(5XA9)5X*PAGE *I3//)

  25  IF (I .EQ. 1 .OR. N LINES .EQ. 0) WRITE(NREPORT,1020) HEADING
1020  FORMAT(34X3A10/
+   *- *7X*SEQ*12X*CARGO ID*11X*CARRIER ID* 13X*TYPE*18X*REF*/)

*           NO MATCH
  30  IF (SEQ TBL(I) .NE. 0) GO TO 40
      WRITE(NREPORT,1001) I, CARG TBL(I), CARR TBL(I), TYPE TBL(I)
1001  FORMAT(5X,I5,5X,3(10X,A10),11X,4H----,5X,12H**NO MATCH**)
      N UNREF = N UNREF + 1
      GO TO 49

*           DUPLICATE RECORD
  40  IF (SEQ TBL(I) .GT. 0) GO TO 45
      IF (SEQ TBL(I) .EQ. -2) GO TO 48
      WRITE(NREPORT,1002) I, CARG TBL(I), CARR TBL(I),
+   TYPE TBL(I)
1002  FORMAT(5X,I5,5X,3(10X,A10),11X,4H----,5X,
+   20H**DUPLICATE RECORD**)
      N DUPL = N DUPL + 1

```


GO TO 49

```
*                               INVALID CARGO IDENT
48      WRITE(NREPORT,1003) I, CARG TBL(I), CARR TBL(I),
+      TYPE TBL(I)
1003    FORMAT(5X,I5,5X,3(10X,A10),11X,4H----,5X,
+      23H**INVALID CARGO IDENT**)
      N INVAL = N INVAL + 1
      GO TO 49
```

```
*                               NO ERRORS
45      WRITE(NREPORT,1000) I, CARG TBL(I), CARR TBL(I),
+      TYPE TBL(I), SEQ TBL(I)
1000    FORMAT(5X,I5,5X,3(10X,A10),11X,I4)
49      NLINES = NLINES + 1
50      CONTINUE
```

```
*                               TABLE INFORMATION SUMMARY
      WRITE(NREPORT,1015) N UNREF, N DUPL, N INVAL
1015    FORMAT(*0*
+      90X35(*-*)/
+      91X15* NON-MATCHES*/
+      91X15* DUPLICATE RECORDS*/
+      91X15* INVALID CARGO IDENTIFIERS*)

      NER ARAY(1) = N UNREF
      NER ARAY(2) = N DUPL
      NER ARAY(3) = N INVAL
```

RETURN

*-----

ENTRY DP RESET

```
*****
*      RESET LINES-PER-PAGE COUNTER      *
*****
```

NLINES = 0

END

```

LOGICAL FUNCTION VLD CG ID(IDENT)
*****
* SEARCH FOR IDENT IN VALID CARGO-IDENT TABLE *
*****
IMPLICIT INTEGER (A-Z)

DIMENSION CARGTBL(650)
COMMON /CGT UNIT/ CT UNIT
COMMON /NREPORT/ NREPORT
DATA PRV CG ID /0/, LMT TABL /0/

SHIFT1R(ID) = ID/2 .A. 37777 77777 77777 77777 B
*-----
* SEE IF IDENT IS IN TABLE
* VALID CARGO IDENTIFIERS
*
VLD CG ID = .FALSE.
IF (LMT TABL .EQ. 0) RETURN

IDTMP = SHIFT1R(IDENT)
5 DO 10 I = 1, LMT TABL
  IF (IDENT .EQ. CARGTBL(I)) GO TO 20
  IF (IDTMP .LT. SHIFT1R(CARGTBL(I))) RETURN
10 CONTINUE
RETURN

20 VLD CG ID = .TRUE.
RETURN
*-----

ENTRY GET CGID
*****
* READ CARGO ID'S FROM FILE ON UNIT NO. (CT UNIT) *
*****
REWIND CT UNIT
VLD CG ID = .FALSE.
DO 100 I = 1, 650
  READ(CT UNIT,1000) CARGTBL(I)
1000 FORMAT(A4)
  IF(EOF(CT UNIT)) 777, 30
  30 IF(SHIFT1R(CARGTBL(I)) .GT. SHIFT1R(PRV CG ID)) GO TO 80
  WRITE(NREPORT,1010) CARGTBL(I), I
1010 FORMAT(*SEQUENCE ERROR IN CARGO IDENT FILE. IDENT= *A4
  + * , ELEMENT NO. *I4)
  RETURN
  80 PRV CG ID = CARGTBL(I)
  100 CONTINUE

WRITE(NREPORT,1020)
1020 FORMAT(*OVERFLOW IN READING CARGO IDENT TABLE.*)
RETURN

```

APPENDIX B

LIGHTER CYCLE TIMES
PROGRAM LISTINGS

```

L5TAG,CM60000,SPUU,T500.ATG
TASK(TN=LOTS,TA=12345,OS=ORISPM,TR=TS,PI=ATG)
COMMENT.*****CTL.STMTS ON SHPDATA4*****
ATTACH,S2K,S2K260,ID=SYS2000,MR=1.
REQUEST,LRDY1,SN=SYS2.
REQUEST,LRDP1,SN=SYS2.
REQUEST,F1,SN=SYS2.
REQUEST,R1,SN=SYS2.
S2K,CR=77.
CATALOG,LRDY1,LRDY14,ID=LOTS.
REWIND,LRDY1.
COPYBF,LRDY1.
CATALOG,LRDP1,LRDP14,ID=LOTS.
REWIND,LRDP1.
COPYBF,LRDP1.
CATALOG,F1,F14,ID=LOTS.
REWIND,F1.
COPYBF,F1.
CATALOG,R1,R14,ID=LOTS.
REWIND,R1.
COPYBF,R1.
*EOR
USER,LOTS;SHARED DBN IS JLMT4A;
REPORT FILE IS LRDY1;
LI/TITLE L(4),L(4),L(4),L(4),L(4),L(4),L(4),L(4)/
C1010,C1110,C1120,C1530,C1510,C1520,C1540,C1920,
OB C1010,C1110,C1120,C1530,C1510,C1520
WH (C1010 EQ TCDF OR C1010 EQ COD)
AND C1910 EQ LRDY;
REPORT FILE IS LRDP1;
LI/TITLE L(4),L(4),L(4),L(4),L(4),L(4),L(4),L(4)/
C1010,C1110,C1120,C1530,C1510,C1520,C1920,
OB C1010,C1110,C1120,C1530,C1510,C1520
WH (C1010 EQ TCDF OR C1010 EQ COD)
AND C1910 EQ LRDP;
REPORT FILE IS F1;
LI/TITLE L(4),L(4),L(4),L(4),L(4),L(4),L(4),L(4)/
C1010,C1110,C1120,C1530,C1510,C1520,
OB C1010,C1110,C1120,C1530,C1510,C1520
WH (C1010 EQ TCDF OR C1010 EQ COD)
AND C1820 EXISTS;
REPORT FILE IS R1;
LI/TITLE L(4),L(4),L(4),L(4),L(4),L(4),L(4),L(4)/
C1010,C1110,C1120,C1530,C1510,C1520,
OB C1010,C1110,C1120,C1530,C1510,C1520
WH (C1010 EQ TCDF OR C1010 EQ COD)
AND C1720 EXISTS;
EXIT;
*EOR
*EOF

```

```

L5TAG,CM60000,SPUU,T500.ATG
TASK(TN=LOTS,TA=12345,OS=ORISPMD,TR=TS,PI=ATG)
COMMENT.*****CTL.STMTS ON SHRDATA4*****
ATTACH,S2K,S2K260,ID=SYS2000,MR=1.
REQUEST,LARR2,SN=SYS2.
REQUEST,LPSN2,SN=SYS2.
REQUEST,LRDY2,SN=SYS2.
REQUEST,LRDP2,SN=SYS2.
S2K,CR=77.
CATALOG,LARR2,LARR24,ID=LOTS.
REWIND,LARR2.
COPYBF,LARR2.
CATALOG,LPSN2,LPSN24,ID=LOTS.
REWIND,LPSN2.
COPYBF,LPSN2.
CATALOG,LRDY2,LRDY24,ID=LOTS.
REWIND,LRDY2.
COPYBF,LRDY2.
CATALOG,LRDP2,LRDP24,ID=LOTS.
REWIND,LRDP2.
COPYBF,LRDP2.
*EOR
USER,LOTS;SHARED DBN IS JLMT4A;
REPORT FILE IS LARR2;
LI/TITLE L(4),L(4),L(4),L(4),L(4),L(4),L(4)/
C1010,C1110,C1120,C1530,C1510,C1520,C1920,
OB C1010,C1110,C1120,C1530,C1510,C1520
WH C1910 EQ LARR AND
(C1010 EQ BBCS OR C1010 EQ ADP OR C1010 EQ ECWY
OR C1010 EQ LACH OR C1010 EQ EDP);
REPORT FILE IS LPSN2;
LI/TITLE L(4),L(4),L(4),L(4),L(4),L(4),L(4)/
C1010,C1110,C1120,C1530,C1510,C1520,C1920,
OB C1010,C1110,C1120,C1530,C1510,C1520
WH C1910 EQ LPSN AND
(C1010 EQ BBCS OR C1010 EQ ADP OR C1010 EQ ECWY
OR C1010 EQ LACH OR C1010 EQ EDP);
REPORT FILE IS LRDY2;
LI/TITLE L(4),L(4),L(4),L(4),L(4),L(4),L(4)/
C1010,C1110,C1120,C1530,C1510,C1520,C1920,
OB C1010,C1110,C1120,C1530,C1510,C1520
WH C1910 EQ LRDY AND
(C1010 EQ BBCS OR C1010 EQ ADP OR C1010 EQ ECWY
OR C1010 EQ LACH OR C1010 EQ EDP);
REPORT FILE IS LRDP2;
LI/TITLE L(4),L(4),L(4),L(4),L(4),L(4),L(4)/
C1010,C1110,C1120,C1530,C1510,C1520,C1920,
OB C1010,C1110,C1120,C1530,C1510,C1520
WH C1910 EQ LRDP AND
(C1010 EQ BBCS OR C1010 EQ ADP OR C1010 EQ ECWY
OR C1010 EQ LACH OR C1010 EQ EDP);
EXIT;
*EOR
*EOF

```

```

L5TAG,CM65000,SFUU,T50.ATG
TASK(TN=LOTS,TA=12345,OS=ORISFMD,TR=TS,PI=ATG)
ATTACH,SHRGEN,SHRGENOBJ,ID=LOTS,MR=1.
ATTACH,LARR2,LARR24,ID=LOTS,MR=1.
ATTACH,LPSN2,LPSN24,ID=LOTS,MR=1.
ATTACH,LRDY2,LRDY24,ID=LOTS,MR=1.
ATTACH,LRDP2,LRDP24,ID=LOTS,MR=1.
SHRGEN.
REWIND,OUTC2.
COPYSBF,OUTC2.
REWIND,OUTI2.
COPYSBF,OUTI2.
REWIND,OUTBN2.
FILE(OUTBN2,MRL=140)
FILE(SORT2)
SORTMRG.
ATTACH,PREV,PREVSHOREOBJ,ID=LOTS.
PREV(SORT2).
ATTACH,READ,UTILITYSHORE24XOBJ,ID=LOTS.
REWIND(OUT).
READ(OUT).
REWIND(OUT).
FILE(OUT,MRL=160)
FILE(SORT3)
SORTMRG.
ATTACH,F,F14,ID=LOTS.
ATTACH,R,R14,ID=LOTS.
REWIND,F.
REWIND,R.
FILE(F,MRL=80,RT=Z,BT=C,FL=80)
FILE(R,MRL=80,RT=Z,BT=C,FL=80)
FILE(FR,MRL=80,RT=Z,BT=C,FL=80)
SORTMRG.
REWIND,FR.
COPYSBF,FR.
ATTACH,SHPGEN,SHPGENOBJ,ID=LOTS,MR=1.
ATTACH,LRDY1,LRDY14,ID=LOTS,MR=1.
ATTACH,LRDP1,LRDP14,ID=LOTS,MR=1.
SHPGEN(,FR).
REWIND,OUTC1.
COPYSBF,OUTC1.
REWIND,OUTI1.
COPYSBF,OUTI1.
REWIND,OUTBN1.
FILE(OUTBN1,MRL=120)
FILE(SORT1)
SORTMRG.
ATTACH,COMBINE,COMBINEOBJ,ID=LOTS.
COMBINE(SORT1,SORT3)
REWIND,OUTC3.

```

```

COPYSBF,OUTC3.
REWIND,OUTI3.
COPYSBF,OUTI3.
REWIND,OUTBN3.
FILE(OUTBN3,MRL=230)
FILE(RES)
SORTMRG.
ATTACH,SHPSHR,SHPSHRRPTOBJ,ID=LOTS.
SHPSHR(RES)
*EOR
SORT
FILE,SORT=OUTBN2,OUTPUT=SORT2
FIELD,FACID(1,10,DISPLAY),DATE(11,10,DISPLAY),SHIFT(21,10,DISPLAY),
,LITTY(31,10,DISPLAY),LITID(41,10,DISPLAY),LITCY(51,4,DISPLAY),
,LARR(61,10,DISPLAY),LPSN(71,10,DISPLAY),LRDY(81,10,DISPLAY),
,LRDP(91,10,DISPLAY),POST(101,10,DISPLAY),RDYT(111,10,DISPLAY),
,CRANE(121,10,DISPLAY),TIME(131,10,DISPLAY)
KEY,FACID(A,DISPLAY),TIME(A,DISPLAY)
END
*EOR
SORT
FILE,SORT=OUT,OUTPUT=SORT3
FIELD,FACID(1,10,DISPLAY),DATE(11,10,DISPLAY),SHIFT(21,10,DISPLAY),
,LITTY(31,10,DISPLAY),LITID(41,10,DISPLAY),LITCY(51,4,DISPLAY),
,LARR(61,10,DISPLAY),LPSN(71,10,DISPLAY),LRDY(81,10,DISPLAY),
,LRDP(91,10,DISPLAY),POST(101,10,DISPLAY),RDYT(111,10,DISPLAY),
,CRANE(121,10,DISPLAY),TIME(131,10,DISPLAY),DIFF(141,10,DISPLAY),
,TYPE(151,10,DISPLAY)
KEY,LITTY(A,DISPLAY),LITID(A,DISPLAY),TIME(A,DISPLAY)
END
*EOR
MERGE
FILE,MERGE=F,R,OUTPUT=FR
FIELD,FACID(4,4,DISPLAY),DATE(11,4,DISPLAY),SHIFT(18,4,DISPLAY),
,LITYP(25,4,DISPLAY),LITID(32,4,DISPLAY),LITCY(39,4,DISPLAY),
,FR(46,4,DISPLAY)
KEY,FACID(A,DISPLAY),DATE(A,DISPLAY),SHIFT(A,DISPLAY),
,LITYP(A,DISPLAY),LITID(A,DISPLAY),LITCY(A,DISPLAY)
END
*EOR
SORT
FILE,SORT=OUTBN1,OUTPUT=SORT1
FIELD,FACID(1,10,DISPLAY),DATE(11,10,DISPLAY),SHIFT(21,10,DISPLAY),
,LITYP(31,10,DISPLAY),LITID(41,10,DISPLAY),LITCY(51,10,DISPLAY),
,LITPS(61,10,DISPLAY),DIR(71,10,DISPLAY),LRDY(81,10,DISPLAY),
,LRDP(91,10,DISPLAY),LSPT(101,10,DISPLAY),TIME(111,10,DISPLAY)
KEY,LITYP(A,DISPLAY),LITID(A,DISPLAY),TIME(A,DISPLAY)
END
*EOR
SORT

```

FILE, SORT=OUTBN3, OUTPUT=RES
FIELD, SHPID(1,10,DISPLAY), DATE(11,10,DISPLAY), SHIFT(21,10,DISPLAY),
, LITTY(31,10,DISPLAY), LITID(41,10,DISPLAY), LITCY(51,10,DISPLAY),
, POST(61,10,DISPLAY), FR(71,10,DISPLAY), LRDY1(81,10,DISPLAY),
, LRDP1(91,10,DISPLAY), DIFF1(101,10,DISPLAY), SHRID(111,10,DISPLAY),
, LARR(121,10,DISPLAY), DIFF2(131,10,DISPLAY), LPSN(141,10,DISPLAY),
, DIFF3(151,10,DISPLAY), LRDY(161,10,DISPLAY), DIFF4(171,10,DISPLAY),
, LRDP(181,10,DISPLAY), DIFF5(191,10,DISPLAY), DIFF6(201,10,DISPLAY),
, LITP(211,10,DISPLAY), TIME(221,10,DISPLAY)
KEY, SHPID(A,DISPLAY), DATE(A,DISPLAY), SHIFT(A,DISPLAY),
, LITTY(A,DISPLAY), LITID(A,DISPLAY), FR(A,DISPLAY),
, TIME(A,DISPLAY)
END
*EOR
*EOF


```

PROGRAM SHPGEN(LRDY1,LRDP1,FR1,OUTI1,OUTC1,OUTPUT,OUTBN1,
+           TAPE1=LRDY1,TAPE2=LRDP1,TAPE3=FR1,
+           TAPE4=OUTI1,TAPE5=OUTC1,TAPE6=OUTPUT,TAPE7=OUTBN1)

C
C   LOGICAL NFIRST,NFINC,NFCOM,NSTOP,NSTOP1,MATCH
C
C   DIMENSION KODE(3),NLRDY(6),NLRDP(6),NFR(6),NPAM(3)
C   DIMENSION NEND(3),NEOF(10)
C
C   COMMON/TBLUNIT/NUNIT1,NUNIT2,NUNIT3,NUNIT4,NUNIT5,NUNIT6,NUNIT7
C
C   INITIALIZE
C   NUNIT1=1
C   NUNIT2=2
C   NUNIT3=3
C   NUNIT4=4
C   NUNIT5=5
C   NUNIT6=6
C   NUNIT7=7
C
C   NFIRST=.FALSE.
C   NFINC=.FALSE.
C   NFCOM=.FALSE.
C   NSTOP=.FALSE.
C   NSTOP1=.FALSE.
C
C   10 CONTINUE
C   SET KODE TO READ THREE FILES.
C   KODE(1)=1
C   KODE(2)=2
C   KODE(3)=3
C
C   READ THREE RECORDS.
C   CALL READ(KODE,NLRDY,NLRDP,NFR,NLGTPS,NPAM,NFIRST,NEND)
C
C   CHECK FOR EOF IN ANY FILE
C   CALL STOP(NEND,NSTOP,NSTOP1,KODE)
C   IF(NSTOP) GO TO 60
C
C   30 CONTINUE
C   COMPARE LRDY AND LRDP RECORDS.
C   CALL COMPARE(NLRDY,NLRDP,MATCH,NFAIL1,NFAIL2)
C
C   CHECK FOR A MATCH
C   IF(MATCH) GO TO 40
C
C   THE MATCH FAILED.  SELECT WHICH FILE TO READ.
C   CALL SELECT(1,NFAIL1,NFAIL2,KODE)
C
C   WRITE THE RECORD THAT DID NOT HAVE A MATCH.

```

```

C      CALL RPTINC(KODE,NLRDY,NLRDP,NFR,NLGTPS,NPAM,NFINC)
C
C      READ A NEW RECORD.
C      CALL READ (KODE,NLRDY,NLRDP,NFR,NLGTPS,NPAM,NFIRST,NEND)
C
C      CHECK FOR EOF IN ANY FILE.
C      CALL STOP(NEND,NSTOP,NSTOP1,KODE)
C      IF(NSTOP) GO TO 60
C
C      START THE LRDY,LRDP COMPARISON OVER.
C      GO TO 30
C
C
C 40 CONTINUE
C      A MATCH WAS FOUND ON THE LRDY AND LRDP RECORDS.
C      COMPARE THE LRDY AND THE F/R RECORDS.
C      CALL COMPARE(NLRDY,NFR,MATCH,NFAIL1,NFAIL2)
C
C      CHECK FOR A MATCH.
C      IF(MATCH) GO TO 50
C
C      THE MATCH FAILED.  SELECT WHICH FILE TO READ.
C      CALL SELECT(2,NFAIL1,NFAIL2,KODE)
C
C      WRITE THE RECORD THAT DID NOT HAVE A MATCH.
C      CALL RPTINC(KODE,NLRDY,NLRDP,NFR,NLGTPS,NPAM,NFINC)
C
C      READ A NEW RECORD.
C      CALL READ(KODE,NLRDY,NLRDP,NFR,NLGTPS,NPAM,NFIRST,NEND)
C
C      CHECK FOR EOF IN ANY FILE
C      CALL STOP(NEND,NSTOP,NSTOP1,KODE)
C      IF(NSTOP) GO TO 60
C
C      IF LRDY AND LRDP RECORDS WERE READ THAN GO TO LRDY,
C      LRDP COMPARE.
C      IF(KODE(1).EQ.1.AND.KODE(2).EQ.2) GO TO 30
C
C      ELSE GO TO LRDY,FR COMPARE
C      GO TO 40
C
C
C 50 CONTINUE
C      A MATCH WAS FOUND ON THE LRDY,LRDP, AND FR RECORDS.
C      CONVERT LRDY TO MINS FROM THE BEGINNING OF THE TEST.
C      CALL CONVERT(NLRDY(2),NLRDY(3),NPAM(1),MINS)
C
C      CONVERT LRDP TO MINS FROM THE BEGINNING OF THE TEST.
C      CALL CONVERT(NLRDY(2),NLRDY(3),NPAM(2),MINS1)
C
C      CALCULATE LIGHTER AT SHIP TIME

```

```

      LSHPT=MINS1-MINS
C
C   WRITE A COMPLETE RECORD.
C   CALL RPTCOM(NLRDY,NLGTPS,NPAM,NFCOM,LSHPT,MINS)
C
C   START OVER WITH READING THREE NEW RECORDS.
C   GO TO 10
C
C
C 60 CONTINUE
C   CHECK FOR EOF IN THREE FILES.
C   IF(NSTOP1) GO TO 70
C
C       WRITE INCOMPLETE RECORDS.
C       CALL RPTINC(KODE,NLRDY,NLRDP,NFR,NLGTPS,NPAM,NFINC)
C       READ A NEW RECORD(S)
C       CALL READ(KODE,NLRDY,NLRDP,NFR,NLGTPS,NPAM,NFIRST,NEND)
C
C       CHECK FOR END CONDITION.
C       CALL STOP(NEND,NSTOP,NSTOP1,KODE)
C
C   CONTINUE STOP PROCESS
C   GO TO 60
C
C
C
C 70 CONTINUE
C
C   WRITE EOF IN BINARY FILE
C   DO 80 J=1,10
C       NEOF(J)=4RZEOF
C 80 CONTINUE
C   WRITE(NUNIT7) NEOF,LSHPT,MINS
C
C   END

```

```

C      SUBROUTINE READ(KODE,NLRDY,NLRDP,NFR,NLGTPS,NPAM,NFIRST,NEND)
C
C      PURPOSE:  READS ONE RECORD FROM ANY OF THREE FILES DEPENDING
C                ON THE VALUE OF KODE
C
C      LOGICAL NFIRST
C
C      DIMENSION KODE(3),NLRDY(6),NLRDP(6),NFR(6),NPAM(3),NEND(3)
C
C      COMMON/TBLUNIT/NUNIT1,NUNIT2,NUNIT3,NUNIT4,NUNIT5,NUNIT6,NUNIT7
C
C      CHECK TO DETERMINE IF THE FILES NEED REWINDING
C      IF(NFIRST) GO TO 10
C          REWIND NUNIT1
C          REWIND NUNIT2
C          REWIND NUNIT3
C          REWIND NUNIT4
C          REWIND NUNIT5
C          REWIND NUNIT7
C          NFIRST=.TRUE.
C
C      10 CONTINUE
C
C      IF KODE EQ 1 THAN READ THE LRDY FILE
C      IF(KODE(1).NE.1) GO TO 30
C          READ(NUNIT1,20) NLRDY,NLGTPS,NPAM(1)
C      20  FORMAT(3X,8(R4,3X))
C          SAVE EOF KODE
C          NEND(1)=NLRDY(1)
C      30 CONTINUE
C
C      IF KODE EQ 2 THAN READ THE LRDP FILE
C      IF(KODE(2).NE.2) GO TO 50
C          READ(NUNIT2,40) NLRDP,NPAM(2)
C      40  FORMAT(3X,7(R4,3X))
C          SAVE EOF KODE
C          NEND(2)=NLRDP(1)
C      50 CONTINUE
C
C      IF KODE EQ 3 THEN READ THE R/F FILE
C      IF(KODE(3).NE.3) GO TO 60
C          READ(NUNIT3,40) NFR,NPAM(3)
C          SAVE EOF KODE
C          NEND(3)=NFR(1)
C      60 CONTINUE
C
C      RETURN
C      END

```

SUBROUTINE COMPARE(NTEST1,NTEST2,MATCH,NFAIL1,NFAIL2)

PURPOSE: DETERMINES IF TWO RECORDS MATCH ON SIX MATCH KEYS.
IF NO MATCH, RETURNS TO THE MAIN PROGRAM THE FIRST
MATCH KEYS WHERE THE MATCH FAILED.

LOGICAL MATCH

DIMENSION NTEST1(6),NTEST2(6)

NTEST1 HAS THE LRDP RECORD.
NTEST2 HAS THE LRDP OR F/R RECORD.
COMPARE ON THE SIX MATCH KEYS.

DO 10 J=1,6

IF(NTEST1(J).EQ.NTEST2(J)) GO TO 10
THE MATCH FAILS.

MATCH=.FALSE.

NFAIL1 HAS THE LRDP MATCH KEY

NFAIL1=NTEST1(J)

NFAIL2 HAS THE LRDP OR F/R MATCH KEY

NFAIL2=NTEST2(J)

RETURN

10 CONTINUE

A MATCH WAS FOUND
MATCH=.TRUE.

RETURN
END

```

SUBROUTINE SELECT(KEY,NFAIL1,NFAIL2,KODE)
C
C PURPOSE: DETERMINES WHICH RECORD TO READ.
C
C DIMENSION KODE(3)
C
C KEY=1 WHEN MATCH FAILED ON LRDY,LRDP RECORDS
C KEY=2 WHEN MATCH FAILED ON LRDY,NF/R RECORDS
C
C GO TO (10,20) KEY
C
10 CONTINUE
C THIS SECTION IS FOR KEY=1. A NON-MATCH BETWEEN A LRDY RECORD
C AND A LRDP RECORD.
C
C TEST TO DETERMINE WHICH FILE (LRDY,LRDP) TO READ.
C IF MATCH KEY LRDY GT MATCH KEY LRDP THEN READ LRDP RECORD.
C ELSE READ LRDY RECORD
C KODE(1)=0
C KODE(2)=2
C KODE(3)=0
C IF(NFAIL1.GT.NFAIL2) GO TO 15
C KODE(1)=1
C KODE(2)=0
C KODE(3)=0
15 CONTINUE
C
C RETURN
C
C
20 CONTINUE
C
C THIS SECTION IS FOR KEY=2. A NON-MATCH BETWEEN LRDY AND NF/R
C FILES.
C
C TEST TO DETERMINE WHICH FILE (LRDY,NF/R) TO READ.
C IF MATCH KEY LRDY IS LT MATCH KEY R/F THAN READ LRDY
C AND LRDP RECORDS.
C ELSE READ R/F RECORD
C KODE(1)=1
C KODE(2)=2
C KODE(3)=0
C IF(NFAIL1.LT.NFAIL2) GO TO 30
C KODE(1)=0
C KODE(2)=0
C KODE(3)=3
30 CONTINUE
C
C RETURN
C END

```

```

C      SUBROUTINE CONVERT(NDAY,NSHFT,NHM,MINS)
C
C      PURPOSE: CONVERTS DAYS,SHIFT,HRS AND MINS TO MINS FROM BEGINNING
C                OF THE TEST
C
C      DECODE(10,10,NDAY) ND
10  FORMAT(6X,I4)
C
C      DECODE(10,10,NSHFT) NS
C
C      DECODE(10,20,NHM) NH,NM
20  FORMAT(6X,I2,I2)
C
C
C      ADD 24 TO HRS LT 18 ON SHIFT 2
      IF(NS.EQ.2.AND.NH.LT.17) NH=NH+24
C
C
C      CALCULATE DAY OF TEST. DAY1 EQ 0.
      ND=ND-218
C
C      CALCULATE MINS.
      MINS=ND*24*60+NH*60+NM
C
C
      RETURN
      END

```

```

C      SUBROUTINE RPTINC(KODE,NLRDY,NLRDP,NFR,NLGTPS,NPAM,NFINC)
C
C      PURPOSE: WRITES AN INCOMPLETE RECORD
C
C      LOGICAL NFINC
C
C      DIMENSION KODE(3),NLRDY(6),NLRDP(6),NFR(6),NPAM(3)
C
C      COMMON/TBLUNIT/NUNIT1,NUNIT2,NUNIT3,NUNIT4,NUNIT5,NUNIT6,NUNIT7
C
C      IF FIRST CALL THAN PRINT HEADER
C      IF(NFINC) GO TO 20
C          XDATE=DATE(JDUMMY)
C          XTIME=TIME(JDUMMY)
C          WRITE(NUNIT4,10) XDATE,XTIME
10      FORMAT(1X,18HINCOMPLETE RECORDS,2(5X,A10),3(5X,R4))
C          NFINC=.TRUE.
20      CONTINUE
C
C      DETERMINE IF THE LRDY RECORD SHOULD BE OUTPUT.
C      IF(KODE(1).NE.1) GO TO 40
C          WRITE(NUNIT4,30) NLRDY,NLGTPS,NPAM(1)
30      FORMAT(1X,4HLRDY,3X,8(R4,3X),I4)
40      CONTINUE
C
C      DETERMINE IF THE LRDP RECORD SHOULD BE OUTPUT.
C      IF(KODE(2).NE.2) GO TO 60
C          WRITE(NUNIT4,50) NLRDP,NPAM(2)
50      FORMAT(1X,4HLRDP,3X,7(R4,3X),I4)
60      CONTINUE
C
C      DETERMINE IF THE F/R RECORD SHOULD BE OUTPUT.
C      IF(KODE(3).NE.3) GO TO 80
C          WRITE(NUNIT4,70) NFR,NPAM(3)
70      FORMAT(1X,4H NFR,3X,7(R4,3X),I4)
80      CONTINUE
C
C      RETURN
C      END

```



```

SUBROUTINE STOP(NEND,NSTOP,NSTOP1,KODE)
C
C PURPOSE: TO DETERMINE IF EOF IN ANY FILE. TO DETERMINE IF
C           EOF IN ALL FILES. SET KODE TO WRITE INCOMPLETE RECORDS.
C
C LOGICAL NSTOP,NSTOP1
C
C DIMENSION NEND(3), KODE(3)
C
C TEST FOR AT LEAST ONE EOF.
C IF(NSTOP) GO TO 60
C   DO 20 J=1,3
C     IF(NEND(J).EQ.4R EOF) GO TO 30
C     NO EOF FOUND
20  CONTINUE
C
C ALL FILES CONTAIN DATA
C NSTOP=.FALSE.
C
C RETURN
C
C
C 30 CONTINUE
C   SET KODE FOR AT LEAST ONE EOF
C   NSTOP=.TRUE.
C
C
C 60 CONTINUE
C   CHECK FOR EOF IN LRDY FILE. SET KODE.
C   KODE(1)=0
C   IF(NEND(1).EQ.4R EOF) GO TO 80
C   KODE(1)=1
C 80 CONTINUE
C
C   CHECK FOR EOF IN LRDP FILE. SET KODE.
C   KODE(2)=0
C   IF(NEND(2).EQ.4R EOF) GO TO 100
C   KODE(2)=2
100 CONTINUE
C
C   CHECK FOR EOF IN THE F/R FILE. SET KODE.
C   KODE(3)=0
C   IF(NEND(3).EQ.4R EOF) GO TO 120
C   KODE(3)=3
120 CONTINUE
C
C   CHECK FOR EOF IN ALL FILES.
C   IF(KODE(1).EQ.0.AND.KODE(2).EQ.0.AND.KODE(3).EQ.0) NSTOP1=.TRUE.
C
C
C RETURN
C END

```

```

PROGRAM SHRGEN(LARR2,LPSN2,LRDY2,LRDP2,OUTI2,OUTC2,OUTBN2,
+           OUTPUT,TAPE1=LARR2,TAPE2=LPSN2,TAPE3=LRDY2,
+           TAPE4=LRDP2,TAPE5=OUTI2,TAPE6=OUTC2,
+           TAPE7=OUTBN2,TAPE8=OUTPUT)

C
LOGICAL NFIRST,NFINC,NFCOM,NSTOP,NSTOP1,MATCH

C
DIMENSION KODE(4),NLARR(7),NLPSN(7),NLRDY(7),NLRDP(7),NTIME(4),
+           NEND(4),NEOF(10)

C
COMMON/TBLUNIT/ NUNIT1,NUNIT2,NUNIT3,NUNIT4,NUNIT5,NUNIT6,NUNIT7,
+           NUNIT8

C
C INITIALIZE
NUNIT1=1
NUNIT2=2
NUNIT3=3
NUNIT4=4
NUNIT5=5
NUNIT6=6
NUNIT7=7
NUNIT8=8

C
NFINC=.FALSE.
NFCOM=.FALSE.
NSTOP=.FALSE.
NSTOP1=.FALSE.
NFIRST=.FALSE.

C
10 CONTINUE
C SET KODE TO READ FOUR FILES.
KODE(1)=1
KODE(2)=2
KODE(3)=3
KODE(4)=4

C
C READ FOUR RECORDS.
CALL READ(KODE,NLARR,NLPSN,NLRDY,NLRDP,NFIRST,NEND)

C
C CHECK FOR EOF IN ANY FILE
CALL STOP(NEND,NSTOP,NSTOP1,KODE)
IF(NSTOP) GO TO 60

C
20 CONTINUE
C COMPARE LARR,LPSN RECORDS
CALL COMPARE(NLARR,NLPSN,MATCH,NFAIL1,NFAIL2)

C
C CHECK FOR A MATCH
IF (MATCH) GO TO 30
C

```

```

C      THE MATCH FAILED.  SELECT WHICH FILE TO READ
C      CALL SELECT(1,NFAIL1,NFAIL2,KODE)
C
C      WRITE THE RECORD THAT DID NOT HAVE A MATCH.
C      CALL RPTINC(KODE,NLARR,NLPSN,NLRDY,NLRDP,NFINC)
C
C      READ A NEW RECORD.
C      CALL READ(KODE,NLARR,NLPSN,NLRDY,NLRDP,NFIRST,NEND)
C
C      CHECK FOR EOF IN ANY FILE
C      CALL STOP(NEND,NSTOP,NSTOP1,KODE)
C      IF(NSTOP) GO TO 60
C
C      START THE LARR,LPSN COMPARISON OVER
C      GO TO 20
C
C
C
C 30 CONTINUE
C      A MATCH WAS FOUND ON THE LARR AND LPSN RECORDS.
C      COMPARE THE LARR AND LRDY RECORDS.
C      CALL COMPARE(NLARR,NLRDY,MATCH,NFAIL1,NFAIL2)
C
C      CHECK FOR A MATCH
C      IF(MATCH) GO TO 40
C
C      THE MATCH FAILED.  SELECT WHICH FILE(S) TO READ.
C      CALL SELECT(2,NFAIL1,NFAIL2,KODE)
C
C      WRITE THE RECORDS THAT DID NOT HAVE A MATCH.
C      CALL RPTINC(KODE,NLARR,NLPSN,NLRDY,NLRDP,NFINC)
C
C      READ NEW RECORDS
C      CALL READ(KODE,NLARR,NLPSN,NLRDY,NLRDP,NFIRST,NEND)
C
C      CHECK FOR EOF IN ANY FILE.
C      CALL STOP(NEND,NSTOP,NSTOP1,KODE)
C      IF(NSTOP) GO TO 60
C
C      IF LARR,LPSN RECORDS WERE READ THAN GO TO LARR,LPSN COMP
C      IF(KODE(1).EQ.1.AND.KODE(2).EQ.2) GO TO 20
C
C      ELSE GO TO LARR,LPSN COMPARE
C      GO TO 30
C
C
C
C 40 CONTINUE
C      A MATCH WAS FOUND ON THE LARR,LPSN,LRDY RECORDS.
C      COMPARE THE LARR AND LRDP RECORDS.

```

```

C      CALL COMPARE(NLARR,NLRDP,MATCH,NFAIL1,NFAIL2)
C
C      CHECK FOR A MATCH.
C      IF(MATCH) GO TO 50
C          THE MATCH FAILED.  SELECT WHICH FILES TO READ.
C          CALL SELECT(3,NFAIL1,NFAIL2,KODE)
C
C          WRITE THE RECORD THAT DID NOT HAVE A MATCH.
C          CALL RPTINC(KODE,NLARR,NLPSN,NLRDY,NLRDP,NFINC)
C
C          READ A NEW RECORD
C          CALL READ(KODE,NLARR,NLPSN,NLRDY,NLRDP,NFIRST,NEND)
C
C          CHECK FOR EOF IN ANY FILE
C          CALL STOP(NEND,NSTOP,NSTOP1,KODE)
C
C          IF LARR,LPSN,LRDY RECORDS WERE READ GO TO LARR,LPSN
C          COMPARE
C          IF(KODE(1).EQ.1.AND.KODE(2).EQ.2.AND.KODE(3).EQ.3)
+      GO TO 20
C
C      ELSE GO TO LARR,LRDP COMPARE
C      GO TO 40
C
C
C
C 50 CONTINUE
C      A MATCH WAS FOUND ON FOUR RECORDS.
C
C      COMPUTE POSITION TIME, READY TIME, LIGHTER AT CRANE TIME, AND
C      MINS FROM START OF THE TEST.
C      CALL COMPUTE(NLARR,NLPSN,NLRDY,NLRDP,NTIME)
C
C      WRITE A COMPLETE REPORT
C      CALL RPTCOM(NLARR,NLPSN,NLRDY,NLRDP,NTIME,NFCOM)
C
C      READ FOUR RECORDS
C      GO TO 10
C
C
C
C 60 CONTINUE
C      CHECK FOR THREE EOFs
C      IF(NSTOP1) GO TO 70
C
C      WRITE INCOMPLETE RECORDS
C      CALL RPTINC(KODE,NLARR,NLPSN,NLRDY,NLRDP,NFINC)
C
C      READ NEW RECORD(S)
C      CALL READ(KODE,NLARR,NLPSN,NLRDY,NLRDP,NFIRST,NEND)

```

```

C
C      CHECK FOR EOF
C      CALL STOP(NEND,NSTOP,NSTOP1,KODE)
C
C      CONTINUE STOP PROCESSE
C      GO TO 60
C
C
C      70 CONTINUE
C
C      WRITE EOF IN BINARY FILE
C      NTIME(4)=99999
C      DO 80 J=1,10
C          NEOF(J)=4RZEOF
C      80 CONTINUE
C      WRITE(NUNIT7) NEOF,NTIME
C
C
C      END

```

```

C      SUBROUTINE READ (KODE,NLARR,NLPSN,NLRDY,NLRDP,NFIRST,NEND)
C
C      PURPOSE: READS ONE RECORD FROM ANY OF FOUR FILES DEPENDING
C               ON THE VALUE OF KODE.
C
C      LOGICAL NFIRST
C
C      DIMENSION KODE(4), NLARR(7), NLPSN(7), NLRDY(7), NLRDP(7), NEND(4)
C
C      COMMON/TBLUNIT/NUNIT1,NUNIT2,NUNIT3,NUNIT4,NUNIT5,NUNIT6,NUNIT7
C      +           ,NUNIT8
C
C      CHECK TO DETERMINE IF THE FILES NEED REWINDING.
C      IF (NFIRST) GOTO 10
C          REWIND NUNIT1
C          REWIND NUNIT2
C          REWIND NUNIT3
C          REWIND NUNIT4
C          REWIND NUNIT5
C          REWIND NUNIT6
C          REWIND NUNIT7
C          NFIRST=.TRUE.
C
C      10 CONTINUE
C
C      IF KODE EQ. 1 THEN READ THE LARR RECORD.
C
C      IF (KODE(1).NE.1) GOTO 30
C          READ(NUNIT1,20) NLARR
C      20  FORMAT(3X,7 (R4,3X))
C          SAVE EOF CARD
C          NEND(1)=NLARR(1)
C      30 CONTINUE
C
C      IF KODE EQ 1 THAN READ THE LPSN FILE.
C      IF (KODE(2) .NE. 2) GOTO 40
C          READ(NUNIT2,20) NLPSN
C          SAVE EOF CODE
C          NEND(2)=NLPSN(1)
C      40 CONTINUE
C
C      IF KODE EQ 3 THEN READ THE LRDY FILE.
C      IF(KODE(3) .NE. 3) GOTO 50
C          READ(NUNIT3,20) NLRDY
C          SAVE EOF CODE
C          NEND(3)=NLRDY(1)
C      50 CONTINUE

```

```
C      IF KODE EQ 4 THAN READ THE LRDP FILE.  
      IF (KODE(4) .NE. 4) GOTO 60  
      READ(NUNIT4,20) NLRDP  
C      SAVE EOF CODE  
      NEND(4)=NLRDP(1)  
60 CONTINUE  
C  
C      RETURN  
      END
```



```

C      SUBROUTINE COMPARE(NTEST1,NTEST2,MATCH,NFAIL1,NFAIL2)
C
C      PURPOSE: DETERMINES IF TWO RECORDS MATCH ON SIX MATCH KEYS.
C                IF NO MATCH, RETURNS TO THE MAIN PROGRAM THE FIRST
C                MATCH KEYS WHERE THE MATCH FAILED.
C
C      LOGICAL MATCH
C
C      DIMENSION NTEST1(7),NTEST2(7)
C
C      NTEST1 HAS THE LARR RECORD.
C      NTEST2 HAS THE LPSN OR LRDY OR LRDP RECORD.
C      COMPARE ON THE SIX MATCH KEYS.
C      DO 10 J=1,6
C          IF(NTEST1(J).EQ.NTEST2(J)) GOTO 10
C
C          THE MATCH FAILS.
C          MATCH=.FALSE.
C          NFAIL1 HAS THE LARR MATCH KEY.
C          NFAIL1=NTEST1(J)
C          NFAIL2 HAS THE LPSN OR LRDY OR LRDP MATCH KEY
C          NFAIL2=NTEST2(J)
C
C          RETURN
C
C      10 CONTINUE
C
C      A MATCH WAS FOUND.
C      MATCH=.TRUE.
C
C      RETURN
C      END

```

```

SUBROUTINE SELECT(KEY,NFAIL1,NFAIL2,KODE)
C
C   DIMENSION KODE(4)
C
C   KEY=1 WHEN MATCH FAILED ON LARR,LPSN, RECORDS.
C   KEY=2 WHEN MATCH FAILED ON LARR,LRDY RECORDS.
C   KEY=3 WHEN MATCH FAILED ON LARR,LRDP RECORDS.
C
C   GOTO (10,30,50) KEY
C
10  CONTINUE
C   THIS SECTION IS FOR KEY=1.  A NON-MATCH BETWEEN A LARR AN
C   LPSN RECORD.
C
C   TEST TO DETERMINE WHICH FILE (LARR,LPSN TO READ. IF MATCH KEY
C   LARR GT MATCH KEY LPSN THEN READ LPSN RECORD.
C   KODE(1)=1
C   KODE(2)=0
C   KODE(3)=0
C   KODE(4)=0
C   NFAIL1 HAD LARR NATCH KEY.  NFAIL2 HAS LPSN MATCH KEY.
C   IF MATCH KEY LARR LT MATCH LEY LPSN THEN READ LARR FILE.
C   IF(NFAIL1.LT.NFAIL2) GOTO 20
C       KODE(1)=0
C       KODE(2)=2
C       KODE(3)=0
C       KODE(4)=0
20  CONTINUE
C
C   RETURN
C
C
30  CONTINUE
C   THIS SECTION IS FOR KEY=2.  A NON-MATCH BETWEEN THE LARR AND
C   THE LRDY
C
C   TEST TO DETERMINE WHICH FILE(S) (LARR,LPSN OR LRDY)
C   TO READ.
C   IF LARR,LPSN MATCH KEY IS LT LRSY MATCH KEY THEN READ LARR
C   LPSN RECORDS.
C   KODE(1)=1
C   KODE(2)=2
C   KODE(3)=0
C   KODE(4)=0
C   NFAIL1 HAS LARR,LPSN MATCH KEY AND NFAIL2 HAS LRDY MATCH KEY.
C   IF(NFAIL1.LT.NFAIL2) GOTO 40
C   IF LARR,LPSN MATCH KEY IS GT LRDY MATCH KEY THEN READ LRDY.
C       KODE(1)=0
C       KODE(2)=0
C       KODE(3)=3
C       KODE(4)=0
40  CONTINUE
C

```

```

      RETURN
C
C
50  CONTINUE
C      THIS SECTION IS FOR KEY=3.  A NON-MATCH BETWEEN THE LARR,LRDP
C      RECORDS.
C
C      TEST TO DETERMINE WHICH FILE(S) TO READ.  IF LARR,LPSN,LRDY
C      MATCH KEY IS LT LRDP MATCH KEY THEN READ LARR,LPSN,LRDY RECORDS.
C      KODE(1)=1
C      KODE(2)=2
C      KODE(3)=3
C      KODE(4)=0
C      NFAIL1 HAS LARR,LPSN,LRDY MATCH KEY.
C      NFAIL2 HAS LRDP MATCH KEY.
C      IF(NFAIL1.LT.NFAIL2) GOTO 60
C      IF LARR,LPSN,LRDY MATCH KEY IS GT LRDP MATCH KEY THEN READ
C      LRDP RECORD.
C      KODE(1)=0
C      KODE(2)=0
C      KODE(3)=0
C      KODE(4)=4
60  CONTINUE
C
C
      RETURN
      END

```

```

C      SUBROUTINE COMPUTE(NLARR,NLPSN,NLRDY,NLRDP,NTIME)
C
C      PURPOSE: COMPUTES LIGHTER POSITION TIME, LIGHTER TIME, AND
C                LIGHTER AT CRANE TIME AND TIME FROM BEGINNING OF TEST.
C
C      DIMENSION NLARR(7),NLPSN(7),NLRDY(7),NLRDP(7),NTIME(4)
C
C      CONVERT LARR TO MINS
C      CALL CONVERT(NLARR,LARR)
C
C      CONVERT NLPSN TO MINS
C      CALL CONVERT (NLRDY,LRDY)
C
C      CONVERT NLPSN TO MINS
C      CALL CONVERT(NLPSN,LPSN)
C
C      CONVERT NLRDP TO MINS
C      CALL CONVERT(NLRDP,LRDP)
C
C      COMPUTE LIGHTER IN POSITION TIME
C      NTIME(1)=LPSN-LARR
C
C      COMPUTE LIGHTER READY TIME
C      NTIME(2)=LRDY-LPSN
C
C      COMPUTE LIGHTER AT CRANE TIME
C      NTIME(3)=LRDP-LRDY
C
C      SAVE TIME FROM BEGINNING OF TEST.
C      NTIME(4)=LRDY
C
C      RETURN
C      END

```

```

C      SUBROUTINE CONVERT(NARRAY,MINS)
C
C      PURPOSE: CONVERTS DAYS,SHIFT,HRS AND MINS TO MINS FROM
C                BEGINNING OF THE TEST.
C
C      DIMENSION NARRAY(7)
C
C      SET DAY
C      NDAY=NARRAY(2)
C
C      SET SHIFT
C      NSHFT=NARRAY(3)
C
C      SET HRS AND MINS
C      NHM=NARRAY(7)
C
C
C      10  DECODE(10,10,NDAY) ND
C          FORMAT(6X,I4)
C
C      DECODE(10,10,NSHFT) NS
C
C      DECODE(10,20,NHM) NH,NM
C      20  FORMAT(6X,I2,I2)
C
C      ADD 24 TO HRS LT 18 ON SHIFT 2
C      IF(NS.EQ.2.AND.NH.LT.17) NH=NH+24
C
C      COMPUTE DAY OF TEST.  DAY 1 EQ 0.
C      ND=ND-218
C
C      COMPUTE MINS
C      MINS= ND*24*60 + NH*60 + NM
C
C      RETURN
C      END

```

```

C      SUBROUTINE RPTINC(KODE,NLARR,NLPSN,NLRDY,NLRDP,NFINC)
C
C      PURPOSE: WRITES AN INCOMPLETE RECORD
C
C      LOGICAL NFINC
C
C      DIMENSION KODE(4),NLARR(7),NLPSN(7),NLRDY(7),NLRDP(7)
C
C      COMMON/TBLUNIT/NUNIT1,NUNIT2,NUNIT3,NUNIT4,NUNIT5,NUNIT6,NUNIT7
C      +           ,NUNIT8
C
C      PRINT HEADER ON FIRST CALL
C      IF(NFINC) GOTO 20
C          XDATE=DATE(JDUMMY)
C          XTIME=TIME(JDUMMY)
C          WRITE(NUNIT5,10) XDATE,XTIME
10      FORMAT(1X,24HINCOMPLETE SHORE RECORDS,2(5X,A10))
C          NFINC=.TRUE.
20      CONTINUE
C
C      DETERMINE IF THE LARR RECORD SHOULD BE OUTPUT.
C      IF(KODE(1).NE.1) GOTO 40
C          WRITE(NUNIT5,30) NLARR
30      FORMAT(1X,4HLARR,7(3X,R4))
40      CONTINUE
C
C      DETERMINE IF THE LPSN RECORD SHOULD BE OUTPUT.
C      IF(KODE(2).NE.2) GOTO 60
C          WRITE(NUNIT5,50) NLPSN
50      FORMAT(1X,4HLPSN,7(3X,R4))
60      CONTINUE
C
C      DETERMINE IF THE LRDY RECORD SHOULD BE OUTPUT.
C      IF(KODE(3).NE.3) GOTO 80
C          WRITE(NUNIT5,70) NLRDY
70      FORMAT(1X,4HLRDY,7(3X,R4))
80      CONTINUE
C
C      DETERMINE IF THE LRDP RECORD SHOULD BE OUTPUT.
C      IF(KODE(4).NE.4) GOTO 100
C          WRITE(NUNIT5,90) NLRDP
90      FORMAT(1X,4HLRDP,7(3X,R4))
100     CONTINUE
C
C      RETURN
C      END

```

```

C      SUBROUTINE RPTCOM(NLARR,NLPSN,NLRDY,NLRDP,NTIME,NFCOM)
C
C      PURPOSE: WRITES A COMPLETE RECORD
C
C      LOGICAL NFCOM
C
C      DIMENSION NLARR(7),NLPSN(7),NLRDY(7),NLRDP(7),NTIME(4)
C
C      COMMON/TBLUNIT/NUNIT1,NUNIT2,NUNIT3,NUNIT4,NUNIT5,NUNIT6,NUNIT7
C      +           ,NUNIT8
C
C      PRINT HEADER ON FIRST CALL
C      IF(NFCOM) GOTO 20
C          XDATE=DATE(JDUMMY)
C          XTIME=TIME(JDUMMY)
C          WRITE(NUNIT6,10) XDATE, XTIME
10      FORMAT(1X,22HCOMPLETE SHORE RECORDS,2(5X,A10))
C          NFCOM=.TRUE.
20      CONTINUE
C
C      WRITE A COMPLETE RECORD
C      WRITE(NUNIT6,30) NLARR,NLPSN(7),NLRDY(7),NLRDP(7),NTIME
30      FORMAT(1X,10(3X,R4),3(3X,I4),3X,I6)
C
C      WRITE(NUNIT7) NLARR,NLPSN(7),NLRDY(7),NLRDP(7),NTIME
C
C
C      RETURN
C      END

```

```

SUBROUTINE STOP(NEND,NSTOP,NSTOP1,KODE)
C
C PURPOSE: TO DETERMINE IF EOF IN ANY FILE.  TO DETERMINE IF EOF
C          IN ALL FILES.  SETS KODE TO WRITE INCOMPLETE RECORDS.
C
C LOGICAL NSTOP,NSTOP1
C
C DIMENSION NEND(4),KODE(4)
C
C TEST FOR AT LEAST ONE EOF
C IF(NSTOP) GO TO 30
C   DO 10 J=1,4
C     IF(NEND (J).EQ.4R EOF) GO TO 20
C   NO EOF FOUND
10  CONTINUE
C
C ALL FILES CONTAIN DATA
C NSTOP=.FALSE.
C
C RETURN
C
20 CONTINUE
C SET KODE FOR AT LEAST ONE EOF
C NSTOP=.TRUE.
C
C
C
30 CONTINUE
C CHECK FOR EOF IN LARR FILE.
C KODE(1)=0
C IF(NEND(1).EQ.4R EOF) GO TO 40
C   KODE(1)=1
40 CONTINUE
C
C CHECK FOR EOF IN LPSN FILE.
C KODE(2)=0
C IF(NEND(2).EQ.4R EOF) GO TO 50
C   KODE(2)=2
50 CONTINUE
C
C CHECK FOR EOF IN LRDY FILE.
C KODE(3)=0
C IF(NEND(3).EQ.4R EOF) GO TO 60
C   KODE(3)=3
60 CONTINUE
C
C CHECK FOR EOF IN LRDY FILE
C KODE(4)=0
C IF(NEND(4).EQ.4R EOF) GO TO 70
C   KODE(4)=4
70 CONTINUE

```


C
C
C

C
C

CHECK FOR EOF IN ALL FILES.
IF(KODE(1).EQ.0.AND.KODE(2).EQ.0.AND.KODE(3).EQ.0
+ .AND.KODE(4).EQ.0) NSTOP1=.TRUE.

RETURN
END

```

C      PROGRAM PREV(TAPE1,OUT,OUTPUT,TAPE2=OUT,TAPE6=OUTPUT)
C
C      PURPOSE: TO SAVE PREVIOUS LIGHTER TYPE AND PREVIOUS LRDP
C                (SHORE) ON CURRENT RECORD.
C
C      DIMENSION N(16)
C
C      NOLD=4RNONE
C
C      DO 20 J=1,9999
C      READ(1) (N(I),I=1,14)
C
C      N(15)=-999
C      N(16)=4RNONE
C      IF(N(1) .NE. NOLD) GO TO 10
C
C          CALL CONVERT(N(2),N(3),N(9),NLRDY)
C          N(15)=NLRDY-NLRDP
C          N(16)=NTYPE
C
C 10  CONTINUE
C
C      WRITE(2) (N(I),I=1,16)
C
C      NTYPE=N(4)
C      NLRDP=N(10)
C      NOLD=N(1)
C
C      IF(N(1) .EQ. 4RZEOF) GO TO 30
C      CALL CONVERT(N(2),N(3),N(10),NLRDP)
C
C 20  CONTINUE
C 30  CONTINUE
C
C      END

```

SUBROUTINE CONVERT(NDAY,NSHFT,NHM,MINS)

PURPOSE: CONVERTS DAYS,SHIFT,HRS AND MINS TO MINS FROM BEGINNING
OF THE TEST

10 DECODE(10,10,NDAY) ND
FORMAT(6X,I4)

DECODE(10,10,NSHFT) NS

20 DECODE(10,20,NHM) NH,NM
FORMAT(6X,I2,I2)

ADD 24 TO HRS LT 18 ON SHIFT 2
IF(NS.EQ.2.AND.NH.LT.17) NH=NH+24

CALCULATE DAY OF TEST. DAY1 EQ 0.
ND=ND-218

CALCULATE MINS.
MINS=ND*24*60+NH*60+NM

RETURN
END

```

PROGRAM COMBINE(SHIP,SHORE,OUTI3,OUTC3,OUTBN3,OUTPUT,
+             TAPE1=SHIP,TAPE2=SHORE,TAPE3=OUTI3,
+             TAPE4=OUTC3,TAPE5=OUTBN3,TAPE6=OUTPUT)

C
C   LOGICAL MISSING,MATCH,NSTOP1,NSTOP,NFWD

C
C   DIMENSION KODE(2),NSHIP(7),NTIME1(4),NSHORE(6),NTIME2(7),NEND(2),
+           ISHIP(7),ITIME1(4),ISHORE(6),ITIME2(7),NTIME3(3)
+           ,ITIME3(3)

C
C   COMMON/TBLUNIT/NUNIT1,NUNIT2,NUNIT3,NUNIT4,NUNIT5,NUNIT6

C
C   INITIALIZE

C
C   NUNIT1=1
C   NUNIT2=2
C   NUNIT3=3
C   NUNIT4=4
C   NUNIT5=5
C   NUNIT6=6

C
C   NSTOP=.FALSE.
C   NSTOP1=.FALSE.
C   MISSING=.FALSE.

C
C   SET KODE TO READ SHIP AND SHORE FILES.

C
C   KODE(1)=1
C   KODE(2)=2

C
10 CONTINUE

C
C   READ A SHIP AND A SHORE RECORD.
C   CALL READ(KODE,NSHIP,NSHORE,NTIME1,NTIME2,NTIME3,NEND)

C
C   CHECK FOR EOF IN ANY FILE
C   CALL STOP(NEND,NSTOP,NSTOP1,KODE)
C   IF(NSTOP) GOTO 150

C
C   COMPARE THE NSHIP AND NSHORE RECORDS ON LIGHTER TYPE AND
C   LIGHTER ID

C
20 CONTINUE
    CALL COMPARE(NSHIP(4),NSHIP(5),NSHORE(4),NSHORE(5),
+             MATCH,NFAIL1,NFAIL2)

C
C   CHECK FOR A MATCH.
C   IF(MATCH) GOTO 30

```

```

C      THE MATCH FAILED.  SELECT WHICH FILE TO READ.
C      CALL SELECT(NFAIL1,NFAIL2,KODE)
C
C      WRITE THE RECORD THAT DID NOT HAVE A MATCH.
C      CALL RPTINC(KODE,NSHIP,NSHORE,NTIME1,NTIME2)
C
C      SET MISSING RECORD KODE.
C      MISSING=.TRUE.
C
C      READ A NEW RECORD.
C      CALL READ(KODE,NSHIP,NSHORE,NTIME1,NTIME2,NTIME3,NEND)
C
C      CHECK FOR EOF IN ANY FILE.
C      CALL STOP(NEND,NSTOP,NSTOP1,KODE)
C      IF(NSTOP) GOTO 150
C
C      START THE NSHIP,NSHORE COMPARISION OVER.
C
C      GOTO 20
C
C      30 CONTINUE
C
C      TWO RECORDS WERE FOUND THAT MATCH ON LIGHTER TYPE AND
C      LIGHTER ID.
C
C      DETERMINE FORWARD OR RETROGRADE MOVEMENT.
C      IF(NTIME1(1).EQ.4R   R) GOTO 50
C
C      THIS SECTION IS FOR FORWARD.
C      NTIME1(2) IS LRDY AT SHIP
C      NTIME2(1) IS LARR AT SHORE
C
C      SET FORWARD CODE EQ TRUE
C
C      NFWD=.TRUE.
C
C      CHECK IF LRDY GT LARR
C      CALL CONVERT(NSHIP(2),NSHIP(3),NTIME1(2),LRDY)
C      CALL CONVERT(NSHORE(2),NSHORE(3),NTIME2(1),LARR)
C      IF(LRDY .GT. LARR) GO TO 40
C      SET KODE TO READ LRDY (SHIP) FILE
C      KODE(1)=1
C      KODE(2)=0
C      CHECK FOR MAX LRDY
C      GOTO 70
C
C      40 CONTINUE
C

```

```

C      WRITE LARR(SHORE) INCOMPLETE RECORD, SET KODE TO READ NEW SHORE
C      RECORD AND SET MISSING RECORD KODE.
C
C      MISSING=.TRUE.
C      KODE(1)=0
C      KODE(2)=2
C
C      CALL RPTINC(KODE,NSHIP,NSHORE,NTIME1,NTIME2)
C
C      READ A NEW LARR (SHORE) RECORD AND START THR COMPARSION OVER.
C      GOTO 10
C
C
C 50 CONTINUE
C      THIS SECTION IS FOR RETROGARDE.
C      SET RETROGRADE CODE.
C      NFWD=.FALSE.
C      NTIME1(2) IS LRDY A SHIP.
C      NTIME2(1) IS LARR AT SHORE.
C
C      CALL CONVERT(NSHORE(2),NSHORE(3),NTIME2(1),LARR)
C      CALL CONVERT(NSHIP(2),NSHIP(3),NTIME1(2),LRDY)
C      IF(LARR .GT. LRDY) GO TO 60
C      ST KODE TO READ LARR (SHORE) FILE.
C      KODE(1)=0
C      KODE(2)=2
C      CHECK FOR MAX LARR
C      GO TO 70
C
C
C 60 CONTINUE
C      WRITE LRDY INCOMPLETE RECORD, SET KODE TO READ NEW LRDY RECORD
C      AND SET MISSING RECORD KODE.
C
C      MISSING=.TRUE.
C
C      KODE(1)=1
C      KODE(2)=0
C
C      CALL RPTINC(KODE,NSHIP,NSHORE,NTIME1,NTIME2)
C
C      READ A NEW LRDY RECORD AND START THE COMPARISION OVER.
C      GOTO 10
C
C
C 70 CONTINUE
C      THIS SECTION CHECKS FOR LRDY MAX OR LARR MAX.
C
C      READ THE LRDY(N+1) OR LARR(N+1) RECORD DEPENDING ON THE VALUE
C      OF KODE.

```

```

C      CALL READ(KODE,ISHIP,ISHORE,ITIME1,ITIME2,ITIME3,NEND)
C
C      CHECK FOR EOF IN ANY FILE
C      CALL STOP(NEND,NSTOP,NSTOP1,KODE)
C      IF(NSTOP) GOTO 150
C
C      IF(NFWD) GOTO 80
C      THIS SECTION IS FOR RETROGRADE
C      CALL COMPARE(NSHIP(4),NSHIP(5),ISHORE(4),ISHORE(5),MATCH,
+      NF1,NF2)
C      GOTO 90
C
C      80 CONTINUE
C      THIS SECTION IS FOR FORWARD
C      CALL COMPARE(ISHIP(4),ISHIP(5),NSHORE(4),NSHORE(5),MATCH,NF1,NF2)
C
C      90 CONTINUE
C
C      CHECK FOR MATCH ON LIGHTER TYPE AND LIGHTER ID
C      IF(MATCH) GOTO 100
C      IF NON-MATCH ACCEPT RECORD
C      GOTO 120
C
C
C      100 CONTINUE
C      A MATCH WAS FOUND.
C      CHECK FOR FORWARD OR RETROGRADE MOVEMENT.
C      IF(.,NOT.NFWD) GO TO 110
C
C      THIS SECTION IS FOR FORWARD.
C      ITIME1(2) EQ LRDY AT SHIP
C      NTIME2(1) EQ LARR AT SHORE
C
C      IF LRDY(N+1) GT LARR(M) ACCEPT RECORD
C      CALL CONVERT(ISHIP(2),ISHIP(3),ITIME1(2),LRDY)
C      CALL CONVERT(NSHORE(2),NSHORE(3),NTIME2(1),LARR)
C      IF(LRDY .GT. LARR) GO TO 120
C
C
C      WRITE AN INCOMPLETE LRDY RECORD
C      CALL RPTINC(KODE,NSHIP,NSHORE,NTIME1,NTIME2)
C
C
C      UPDATE CURRENT LRDY RECORD
C      CALL UPDATE(KODE,ISHIP,NSHIP,ISHORE,NSHORE,ITIME1,NTIME1,
+      ITIME2,NTIME2,ITIME3,NTIME3)
C
C      SET MISSING RECORD CODE.
C      MISSING=.TRUE.
C
C      READ NEW LRDY RECORD
C      GOTO 70

```

```

C
C 110 CONTINUE
C   THIS SECTION IS FOR RETROGRADE
C   NTIME1(2) EQ LRDY AT SHIP
C   ITIME2(2) EQ LARR AT SHORE
C
C   IF LARR(N+1) GT LRDY(M) ACCEPT RECORD
C   CALL CONVERT(ISHORE(2),ISHORE(3),ITIME2(1),LARR)
C   CALL CONVERT(NSHIP(2),NSHIP(3),NTIME1(2),LRDY)
C   IF(LARR .GT. LRDY) GO TO 120
C
C       WRITE AN INCOMPLETE LARR RECORD
C       CALL RPTINC(KODE,NSHIP,NSHORE,NTIME1,NTIME2)
C
C       UPDATE CURRENT LARR RECORD
C       CALL UPDATE(KODE,ISHIP,NSHIP,ISHORE,NSHORE,ITIME1,NTIME1,--
+           ITIME2,NTIME2,ITIME3,NTIME3)
C
C       SET MISSING CODE
C       MISSING=.TRUE.
C
C   READ NEW LARR RECORD
C   GOTO 70
C
C
C 120 CONTINUE
C   A RECORD HAS BEEN COMBINED.
C
C   COMPUTE LIGHTER SUCCESSION TIME AND LIGHTER TRANSIT TIME
C   CALL COMPUTE(NSHIP,NSHORE,NTIME1,NTIME2,NFWD,MISSING,
+       NTIME3)
C
C   INITIALIZE MISSING RECORD CODE
C   MISSING=.FALSE.
C   CALL CONVERT(NSHIP(2),NSHIP(3),NTIME1(2),NSORT)
C
C
C   WRITE A COMPLETE RECORD
C   CALL RPTCOM(NSHIP,NSHORE,NTIME1,NTIME2,NTIME3,NSORT)
C
C   SET KODE TO READ NEW RECORDS
C   IF(NFWD) GOTO 130
C   SET KODE FOR RETROGRADE
C   KODE(1)=0
C   KODE(2)=0
C   CALL UPDATE(KODE,ISHIP,NSHIP,ISHORE,NSHORE,ITIME1,
+       NTIME1,ITIME2,NTIME2,ITIME3,NTIME3)
C
C   KODE(1)=1
C   KODE(2)=0

```



```

      GOTO 140
C
130 CONTINUE
C   SET KODE FOR FORWARD
      KODE(1)=1
      KODE(2)=0
C
      CALL UPDATE(KODE,ISHIP,NSHIP,ISHORE,NSHORE,ITIME1,NTIME1,
+             ITIME2,NTIME2,ITIME3,NTIME3)
      KODE(1)=0
      KODE(2)=2
140 CONTINUE
C   GOTO BEGINNING OF PROGRAM
      GOTO 10
C
C
C
150 CONTINUE
C   CHECK FOR EOF IN TWO FILES
      IF(NSTOP1) GOTO 160
C
C       WRITE AN INCOMPLETE RECORD
      CALL RPTINC(KODE,NSHIP,NSHORE,NTIME1,NTIME2)
C
C       READ A NEW RECORD
      CALL READ(KODE,NSHIP,NSHORE,NTIME1,NTIME2,NTIME3,NEND)
C
C       CHECK FOR EOF IN BOTH FILES.
      CALL STOP(NEND,NSTOP,NSTOP1,KODE)
C
C   CONTINUE STOP PROCESS
      GOTO 150
C
C
C
160 CONTINUE
C
C   WRITE EOF IN BINARY FILE
      DO 170 J=1,7
          NSHIP(J)=4RZEOF
          NTIME2(J)=9999
170 CONTINUE
      WRITE(NUNIT5) NSHIP,(NSHIP(J),J=1,3),NTIME2(1),NSHIP(1),
+      (NSHIP(J),NTIME2(J),J=1,4),NTIME2(1),NSHIP(1)
C
      END

```

```

C      SUBROUTINE READ (KODE,NSHIP,NSHORE,NTIME1,NTIME2,NTIME3,NEND)
C
C      PURPOSE: TO READ A RECORD FROM THE SHIP AND/OR SHORE FILE
C
C      LOGICAL FIRST
C
C      DIMENSION KODE(2),NSHIP(7),NSHORE(6),NTIME1(4),NTIME2(7),NEND(2)
C      +          ,NTIME3(3)
C
C      COMMON/TBLUNIT/NUNIT1,NUNIT2,NUNIT3,NUNIT4,NUNIT5,NUNIT6
C
C      DATA FIRST/6H.TRUE./
C
C      REWIND FILES FIRST TIME THRU
C      IF(.NOT. FIRST) GO TO 10
C          REWIND NUNIT1
C          REWIND NUNIT2
C          REWIND NUNIT3
C          REWIND NUNIT4
C          REWIND NUNIT5
C          FIRST = .FALSE.
10  CONTINUE
C      IF KODE(1)=1 READ SHIP FILE
C      IF( KODE(1) .NE. 1) GO TO 20
C      READ(NUNIT1) NSHIP,NTIME1
C          NEND(1)=NSHIP(1)
20  CONTINUE
C      IF KODE(2)=2 READ SHORE FILE
C      IF(KODE(2).NE.2) GO TO 30
C      READ(NUNIT2) NSHORE,NTIME2,NSPACE,NTIME3(2),NTIME3(3)
C          NEND(2)=NSHORE(1)
30  CONTINUE
C      RETURN
C      END

```

HD-A131 715

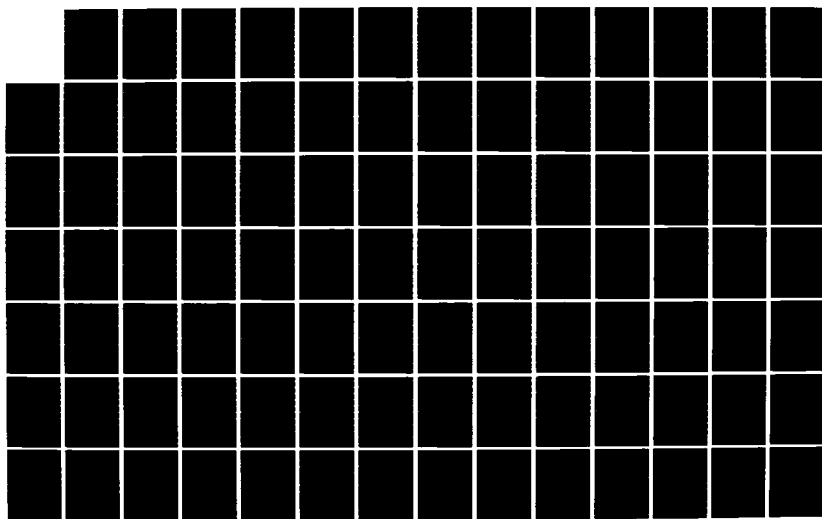
JOINT LOGISTICS-OVER-THE-SHORE (LOTS) MAIN TEST
AUTOMATED DATA BASE REDUCTION PROGRAMS(U) ORI INC
SILVER SPRING MD H CASEY ET AL. MAR 79 ORI-TR-1477
MDA903-75-C-0016

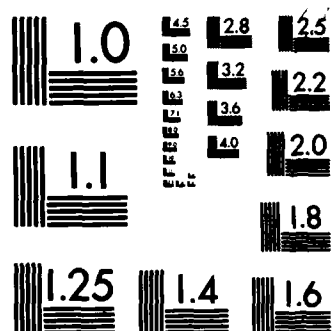
4/6

UNCLASSIFIED

F/G 9/2

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```

SUBROUTINE COMPARE(N1,N2,N3,N4
+               ,MATCH,NFAIL1,NFAIL2)
C
C
C   PURPOSE: DETERMINES IF TWO RECORDS MATCH ON TWO MATCH KEYS
C             (LIGHTER TYPE AND LIGHTER ID).
C
C   LOGICAL MATCH
C
C   DIMENSION NTEST1(2),NTEST2(2)
C
C   NTEST1(1)=N1
C   NTEST1(2)=N2
C   NTEST2(1)=N3
C   NTEST2(2)=N4
C   NTEST1 HAS NSHIP KEYS
C   NTEST2 HAS NSHORE KEYS
C
C   DO 10 J=1,2
C       IF(NTEST1(J).EQ.NTEST2(J)) GOTO 10
C       THE MATCH FAILS
C       MATCH=.FALSE.
C       NFAIL1 HAS THE NSHIP MATCH KEY
C       NFAIL1=NTEST1(J)
C       NFAIL2 HAS THE NSHORE MATCH KEY
C       NFAIL2=NTEST2(J)
C
C       RETURN
C
C 10 CONTINUE
C
C   A MATCH HAS BEEN FOUND
C   MATCH=.TRUE.
C
C   RETURN
C   END

```

```

SUBROUTINE SELECT(NFAIL1,NFAIL2,KODE)
C
C
C   PURPOSE: DETERMINES WHICH RECORD TO READ
C
C   DIMENSION KODE(2)
C
C   TEST TO DETERMINE WHICH FILE (NSHIP,NSHORE) TO READ.
C   KODE(1)=1
C   KODE(2)=0
C
C   NFAIL1 HAS NSHIP MATCH KEY.
C   NFAIL2 HAS NSHORE MATCH KEY.
C   IF NSHIP MATCH KEY LT NSHORE MATCH KEY THEN READ NSHIP RECORD.
C   IF(NFAIL1.LT.NFAIL2) GOTO 10
C       KODE(1)=0
C       KODE(2)=2
C
C
C   10 CONTINUE
C
C
C   RETURN
C   END

```

SUBROUTINE COMPUTE(NSHIP,NSHORE,NTIME1,NTIME2,NFWD,MISSING,
+NTIME3)

PURPOSE: TO COMPUTE LIGHTER SUCCESSION TIME AT SHORE.
TO COMPUTE LIGHTER UNDER WAY TIME
TO SAVE PREVIOUS LIGHTER TYPE

LOGICAL MISSING,NFWD

DIMENSION NSHIP(7),NSHORE(6),NTIME1(4),NTIME2(7),NTIME3(3)

IF(.NOT.MISSING) GO TO 10
MISSING=.TRUE.

10 CONTINUE

IF(NFWD) GO TO 20

CALCULATE LIGHTER READY AT SHIP TIME.

CALL CONVERT(NSHIP(2),NSHIP(3),NTIME1(2),LRDY)

CALCULATE LIGHTER READY TO DEPART SHORE TIME.

CALL CONVERT(NSHORE(2),NSHORE(3),NTIME2(4),LRDP)

CALCULATE LIGHTER UNDER WAY TIME

NTIME3(1)=LRDY-LRDP

GO TO 30

20 CONTINUE

CALCULATE LIGHTER ARRIVAL AT THE SHORE TIME

CALL CONVERT(NSHORE(2),NSHORE(3),NTIME2(1),LARR)

CALCULATE LIGHTER READY TO DEPART SHIP TIME.

CALL CONVERT(NSHIP(2),NSHIP(3),NTIME1(3),LRDP)

CALCULATE LIGHTER UNDERWAY TIME FOR FORWARD

NTIME3(1)=LARR-LRDP

30 CONTINUE

RETURN
END

SUBROUTINE CONVERT(NDAY,NSHFT,NHM,MINS)

PURPOSE: CONVERTS DAYS,SHIFT,HRS AND MINS TO MINS FROM BEGINNING
OF THE TEST

SEPARATE NHM INTO HOURS AND MINS.

10 DECODE(10,10,NDAY) ND
10 FORMAT(6X,I4)

DECODE(10,10,NSHFT) NS

20 DECODE(10,20,NHM) NH,NM
20 FORMAT(6X,I2,I2)

ADD 24 TO HRS LT 18 ON SHIFT 2
IF(NS.EQ.2,AND.NH.LT.17) NH=NH+24

CALCULATE DAY OF TEST. DAY1 EQ 0.
ND=ND-218

CALCULATE MINS.
MINS=ND*24*60 + NH*60 + NM

RETURN
END


```

SUBROUTINE UPDATE(KODE,NSHIP,LSHIP,NSHORE,LSHORE,NTIME1,LTIME1,
+               NTIME2,LTIME2,NTIME3,LTIME3)
C
C   PURPOSE: TO SAVE CURRENT RECORD DEPENDING ON VALUE OF KODE.
C
C   DIMENSION KODE(2),NSHIP(7),LSHIP(7),NSHORE(6),LSHORE(6),
+           NTIME1(4),LTIME1(4),NTIME2(7),LTIME2(7)
+           ,NTIME3(3),LTIME3(3)
C
C   IF KODE=1 SAVE SHIP RECORD
IF(KODE(1).NE.1) GOTO 20
  DO 10 J=1,7
    LSHIP(J)=NSHIP(J)
10  CONTINUE
C
    DO 15 J=1,4
      LTIME1(J)=NTIME1(J)
15  CONTINUE
    GOTO 50
C
20 CONTINUE
C   ELSE SAVE SHORE RECORD
  DO 30 J=1,6
    LSHORE(J)=NSHORE(J)
30 CONTINUE
C
    DO 40 J=1,7
      LTIME2(J)=NTIME2(J)
40 CONTINUE
C
    DO 45 J=1,3
      LTIME3(J)=NTIME3(J)
45 CONTINUE
C
50 CONTINUE
C
C   RETURN
END

```

```

C      SUBROUTINE RPTINC (KODE,NSHIP,NSHORE,NTIME1,NTIME2)
C
C      PURPOSE: TO WRITE A NON-MATCHING RECORD
C
C      LOGICAL FIRST
C
C      DIMENSION KODE(2),NSHIP(7),NSHORE(6),NTIME1(4),NTIME2(7)
C
C      COMMON/TBLUNIT/NUNIT1,NUNIT2,NUNIT3,NUNIT4,NUNIT5,NUNIT6
C
C      DATA FIRST/6H.TRUE./
C
C      IF(.NOT.FIRST) GOTO 10
C          WRITE(NUNIT3,100)
100     FORMAT(1H1,20X,30HNONMATCHING SHIP/SHORE RECORDS)
C          FIRST=.FALSE.
C      10 CONTINUE
C          IF(KODE(2).EQ.2) GOTO 20
C          WRITE(NUNIT3,110) NSHIP,NTIME1
110     FORMAT(5X,6HSHIP ,10(R4,1X),1X,I4,1X,I6)
C          RETURN
C      20 CONTINUE
C          WRITE(NUNIT3,120) NSHORE,NTIME2
120     FORMAT( 5X, 6HSHORE , 10(R4,1X), 3(I4,1X), I6)
C
C      RETURN
C      END

```

```

C      SUBROUTINE RPTCOM(NSHIP,NSHORE,NTIME1,NTIME2,NTIME3,NSORT)
C
C      PURPOSE: TO WRITE A COMBINED RECORD BOTH IN BINARY AND BCD
C      LOGICAL FIRST
C
C      DIMENSION NSHIP(7),NSHORE(6),NTIME1(4),NTIME2(7),NTIME3(3)
C
C      COMMON/TBLUNIT/NUNIT1,NUNIT2,NUNIT3,NUNIT4,NUNIT5,NUNIT6
C
C      DATA FIRST/.TRUE./
C
C      IF(.NOT.FIRST) GOTO 10
C        WRITE(NUNIT4,100)
100    FORMAT(10X, 32HLIGHTER SHIP TO SHORE TIMES FILE      )
C        FIRST=.FALSE.
C      10 CONTINUE
C
C        WRITE(NUNIT4,110)  NSHIP,(NTIME1(I),I=1,4),NSHORE(1),NTIME2(1),
C        +                  NTIME3(1),
C        +                  NTIME2(2),NTIME2(5),NTIME2(3),NTIME2(6),
C        +                  NTIME2(4),
C        +                  NTIME2(7),NTIME3(2),NTIME3(3)
110    FORMAT(3X, 10(R4,1X), 1X, I4, 1X,R4,1X,4(R4,1X,I4,1X), I4,1X,R4)
C
C        WRITE(NUNIT5)      NSHIP,(NTIME1(I),I=1,4),NSHORE(1),NTIME2(1),
C        +                  NTIME3(1),
C        +                  NTIME2(2),NTIME2(5),NTIME2(3),NTIME2(6),
C        +                  NTIME2(4),
C        +                  NTIME2(7),NTIME3(2),NTIME3(3),NSORT
C
C      RETURN
C      END

```

```

SUBROUTINE STOP(NEND,NSTOP,NSTOP1,KODE)
C
C PURPOSE: TO DETERMINE IF EOF IS IN ANY FILE. TO DETERMINE IF
C           EOF IS IN ALL FILES. SETS KODE TO WRITE INCOMPLETE
C           RECORDS.
C
C LOGICAL NSTOP,NSTOP1
C
C DIMENSION NEND(2),KODE(2)
C
C TEST FOR AT LEAST ONE EOF
C IF(NSTOP) GOTO 30
C   DO 10 J=1,2
C     IF(NEND(J).EQ.4RZEOF) GOTO 20
C     NO EOF FOUND
10 CONTINUE
C
C ALL FILES CONTAIN DATA
C NSTOP=.FALSE.
C
C RETURN
C
20 CONTINUE
C SET KODE FOR A LEAST ONE EOF
C NSTOP=.TRUE.
C
30 CONTINUE
C CHECK FOR EOF IN SHIP FILE.
C KODE(1)=0
C IF(NEND(1).NE.4RZEOF) KODE(1)=1
C
C CHECK FOR EOF IN NSHORE FILE.
C KODE(2)=0
C IF(NEND(2).NE.4RZEOF) KODE(2)=2
C
C CHECK FOR EOF IN ALL FILES.
C IF(KODE(1).EQ.0.AND.KODE(2).EQ.0) NSTOP1=.TRUE.
C
C RETURN
C END

```

```

C
C
C
C
C
C
PROGRAM WTRPT(TAPE1,OUTPUT,TAPE6=OUTPUT)

C
C
C
C
C
C
PURPOSE: TO READ THE SORTED SHIP TO SHORE TIMES FILE, COMPUTE
          STATISTICS AND WRITE REPORT. FOR EACH SHIP FACILITY, DATE
          AND SHIFT, STATISTICS ARE ACCUMULATED ON INDIVIDUAL
          LIGHTERS AND LIGHTER TYPES.

C
C
C
C
C
C
LOGICAL FIRST,KPGBR
DIMENSION NSHIP(11),NSHORE(11),SUMID(6),SUMSQID(6),
2 SUMTY(6,2), SUMSQTY(6,2),NOID(3),NOTYF(3),NOTYR(3)
DATA BLANK/1H /
FIRST= .TRUE.

C
C
C
C
C
C
INITIALIZE BREAK KEYS AND STATISTICS

OFAC=BLANK
ODATE=BLANK
OSHOFT=BLANK
20 CONTINUE

C
C
C
C
C
C
READ A SHIP TO SHORE TIMES RECORD

READ(1) NSHIP,NSHORE
IF(NSHIP(1) .NE. 4RZEOF) GO TO 40
CALL PRNIST( SUMID,SUMSQID,NOID)
CALL PRNTST( SUMTY,SUMSQTY,NOTYF,NOTYR)
STOP
40 CONTINUE
KPGBR= .FALSE.

C
C
C
C
C
C
CHECK FOR PAGE BREAK ON SHIP FAC,DATE OR SHIFT

IF((NSHIP(1) .NE. OFAC) .OR. (NSHIP(2) .NE. ODATE) .OR.
1 (NSHIP(3) .NE. OSHOFT) ) CALL PGBRK(FIRST,OFAC,ODATE,OSHOFT,
2 NSHIP,NSHORE,KPGBR,OLTRTY,OLTRID,OFR,
3 SUMID,SUMSQID,SUMTY,SUMSQTY,
4 NOID,NOTYF,NOTYR)
IF(KPGBR) GO TO 20

C
C
C
C
C
C
CHECK FOR LIGHTER TYPE,LIGHTER ID,F OR R BREAK

IF( NSHIP(4) .EQ. OLTRTY ) GO TO 50
OLTRTY=NSHIP(4)
OLTRID=NSHIP(5)
OFR=NSHIP(8)
C
CALL PRNIST( SUMID,SUMSQID,NOID)
CALL PRNTST( SUMTY,SUMSQTY,NOTYF,NOTYR)
CALL INID(SUMID,SUMSQID,NOID)
CALL INTY(SUMTY,SUMSQTY,NOTYF,NOTYR)

```

```

CALL ACCID(SUMID,SUMSQID,NOID,NSHIP,NSHORE)
CALL ACCTY(SUMTY,SUMSQTY,NOTYF,NOTYR,NSHIP,NSHORE)
CALL PRNIND(NSHIP,NSHORE)
GO TO 20
C 50 IF((NSHIP(5) .EQ. OLTRID) .AND. (NSHIP(8) .EQ. OFR)) GO TO 60
50 IF(NSHIP(8) .EQ. OFR) GO TO 60
C    CALL PRNIST( SUMID,SUMSQID,NOID)
    OLTRID=NSHIP(5)
    OFR=NSHIP(8)
    CALL INID(SUMID,SUMSQID,NOID)
CALL ACCID(SUMID,SUMSQID,NOID,NSHIP,NSHORE)
CALL ACCTY(SUMTY,SUMSQTY,NOTYF,NOTYR,NSHIP,NSHORE)
CALL PRNIND(NSHIP,NSHORE)
GO TO 20
60 CONTINUE
CALL PRNIND(NSHIP,NSHORE)
CALL ACCID(SUMID,SUMSQID,NOID,NSHIP,NSHORE)
CALL ACCTY(SUMTY,SUMSQTY,NOTYF,NOTYR,NSHIP,NSHORE)
GO TO 20
END

```

C
C
C

SUBROUTINE ACCID(SUMID,SUMSQID,NOID,NSHIP,NSHORE)
DIMENSION SUMID(6),SUMSQID(6),NSHIP(11),NSHORE(11)

ACCUMULATE INDIVIDUAL LIGHTER STATISTICS

```

DIMENSION NOID(3)
NOID(1)=NOID(1) + 1
SUMID(1)      = NSHIP(11) + SUMID(1)
IF(NSHORE(3).EQ.-999) GOTO 10
NOID(2)=NOID(2) + 1
SUMID(2)      = NSHORE(3) + SUMID(2)
SUMSQID(2)    = NSHORE(3) *NSHORE(3) + SUMSQID(2)
10 CONTINUE
SUMID(3)      = NSHORE(5) + SUMID(3)
SUMID(4)      = NSHORE(7) + SUMID(4)
SUMID(5)      = NSHORE(9) + SUMID(5)
IF(NSHORE(10).EQ.-999) GOTO 20
NOID(3)=NOID(3) + 1
SUMID(6)      = NSHORE(10)+ SUMID(6)
SUMSQID(6)    = NSHORE(10)*NSHORE(10) + SUMSQID(6)
20 CONTINUE
SUMSQID(1)    = NSHIP(11)*NSHIP(11) + SUMSQID(1)
SUMSQID(3)    = NSHORE(5) *NSHORE(5) + SUMSQID(3)
SUMSQID(4)    = NSHORE(7) *NSHORE(7) + SUMSQID(4)
SUMSQID(5)    = NSHORE(9) *NSHORE(9) + SUMSQID(5)
RETURN
END

```

```

C      SUBROUTINE ACCTY(SUMTY,SUMSQTY,NOTYF,NOTYR,NSHIP,NSHORE)
C
C      ACCUMULATE LIGHTER TYPE STATISTICS
C      DIMENSION SUMTY(6,2),SUMSQTY(6,2),NSHIP(11),NSHORE(11),
C      2 NOTYR(3),NOTYF(3)
C
C      DATA BLANK/1H /
C      DATA NEFF/4R   F/
C      IF( NSHIP(8) .EQ. NEFF ) GO TO 80
C
C      RETROGRADE LIGHTER TYPE STATISTICS
C
C      NOTYR(1)= NOTYR(1)+1
C      SUMTY(1,2)      = NSHIP(11) + SUMTY(1,2)
C      IF(NSHORE(3).EQ.-999) GOTO 10
C      NOTYR(2)=NOTYR(2) + 1
C      SUMSQTY(2,2)     =NSHORE(3)*NSHORE(3) + SUMSQTY(2,2)
C      SUMTY(2,2)      = NSHORE(3) + SUMTY(2,2)
10  CONTINUE
C      SUMTY(3,2)      = NSHORE(5) + SUMTY(3,2)
C      SUMTY(4,2)      = NSHORE(7) + SUMTY(4,2)
C      SUMTY(5,2)      = NSHORE(9) + SUMTY(5,2)
C      IF(NSHORE(10).EQ.-999) GOTO 20
C      NOTYR(3)=NOTYR(3) + 1
C      SUMSQTY(6,2)=NSHORE(10)*NSHORE(10) +SUMSQTY(6,2)
C      SUMTY(6,2)      = NSHORE(10)+ SUMTY(6,2)
20  CONTINUE
C      SUMSQTY(1,2)     = NSHIP(11) *NSHIP(11) + SUMSQTY(1,2)
C      SUMSQTY(3,2)     = NSHORE(5) *NSHORE(5) + SUMSQTY(3,2)
C      SUMSQTY(4,2)     = NSHORE(7) *NSHORE(7) + SUMSQTY(4,2)
C      SUMSQTY(5,2)     = NSHORE(9) *NSHORE(9) + SUMSQTY(5,2)
C      RETURN
80  CONTINUE
C
C      FORWARD LIGHTER TYPE STATISTICS
C
C      NOTYF(1) = NOTYF(1) + 1
C      IF(NSHORE(3).EQ.-999) GOTO 30
C      NOTYF(2)=NOTYF(2) + 1
C      SUMSQTY(2,1)     =NSHORE(3)*NSHORE(3) + SUMSQTY(2,1)
C      SUMTY(1,1)      = NSHIP(11) + SUMTY(1,1)
C      SUMTY(2,1)      = NSHORE(3) + SUMTY(2,1)
30  CONTINUE

```



```

SUMTY(3,1)      = NSHORE(5) + SUMTY(3,1)
SUMTY(4,1)      = NSHORE(7) + SUMTY(4,1)
SUMTY(5,1)      = NSHORE(9) + SUMTY(5,1)
IF(NSHORE(10).EQ.-999) GOTO 40
NOTYF(3)=NOTYF(3) + 1
SUMSQTY(6,1)     =NSHORE(10)*NSHORE(10) + SUMSQTY(6,1)
SUMTY(6,1)      = NSHORE(10)+ SUMTY(6,1)
40 CONTINUE
SUMSQTY(1,1)     = NSHIP(11) *NSHIP(11) + SUMSQTY(1,1)
SUMSQTY(3,1)     = NSHORE(5) *NSHORE(5) + SUMSQTY(3,1)
SUMSQTY(4,1)     = NSHORE(7) *NSHORE(7) + SUMSQTY(4,1)
SUMSQTY(5,1)     = NSHORE(9) *NSHORE(9) + SUMSQTY(5,1)
RETURN
END

```

```
SUBROUTINE INID(SUMID,SUMSQID,NOID)
  DIMENSION SUMID(6), SUMSQID(6),NOID(3)
  NOID(1)=0
  NOID(2)=0
  NOID(3)=0
  DO 10 I=1,6
    SUMID(I) =0.
    SUMSQID(I)=0.
10 CONTINUE
  RETURN
  END
```

```

SUBROUTINE INTY(SUMTY,SUMSQTY,NOTYF,NOTYR)
DIMENSION SUMTY(6,2),SUMSQTY(6,2),NOTYF(3),NOTYR(3)
DO 5 I=1,3
NOTYF(I)=0
NOTYR(I)=0
5 CONTINUE
DO 10 I=1,6
DO 20 J=1,2
SUMTY(I,J)=0.
SUMSQTY(I,J)=0.
20 CONTINUE
10 CONTINUE
RETURN
END

```

```

SUBROUTINE PGBRK(FIRST,OFAC,ODATE,OSHIPT,NSHIP,NSHORE,KPGBR,
1 OLTRTY,OLTRID,OFR,SUMID,SUMSQID,SUMTY,SUMSQTY,NOID,NOTYF,
2 NOTYR)
  LOGICAL FIRST, KPGBR
  DIMENSION NSHIP(11),NSHORE(11),SUMID(6),SUMSQID(6),SUMTY(6,2),
1 SUMSQTY(6,2),NOID(3),NOTYF(3),NOTYR(3)
  IF(FIRST) GO TO 10
C   PRINT LIGHTER ID AND TYPE STATISTICS
C   CALL PRNIST(SUMID,SUMSQID,NOID)
C   CALL PRNTST(SUMTY,SUMSQTY,NOTYF,NOTYR)
C   NEW PAGE HEADING
10  CONTINUE
  FIRST = .FALSE.
C   COMPUTE DAY OF AUGUST
  DECODE(10,20,NSHIP(2)) ND
20  FORMAT(6X,I4)
  ND=ND-212
  WRITE(6,100) NSHIP(1),ND,NSHIP(3)
100 FORMAT(1H1, 10X, 27HLIGHTER SHIP TO SHORE TIMES / 3X,18HSHIP FACI
1LITY ID: R4, 3X, 9HDATE: AUG I3, 3X, 6HSHIFT: R4, / )
  WRITE(6,110)
110 FORMAT( 21X,4HFWD 11X,4HLTR 1X,4HSHOR 6X,4HLTR 6X,4HLPSN6X,4HLRDY
16X,3HLTR7X,4HPREV/1X,3HLTR2X,3HLTR2X, 2(3HLTR2X,), 4H OR 11X,
2 4H AT 1X,4HFAC 6X,4HUNDR 6X,4H - 6X,4H - 6X,4H AT 1X, 4HLTR
3 1X, 4HLTR / 1X,4HTYPE 1X,4H ID 1X,4HCYCL 1X,4HPOS. 1X,4HRETR 1X,
4 4HLRDY 1X,4HLRDP 1X,4HSHIP1X,4H ID 1X,4HLARR 1X,4HWAY 1X,4HLPSN
5 1X,4HLARR,
6 1X,4HLRDY 1X,4HLPSN 1X,4HLRDP 1X,4HCRAN 1X,4HSUCC 1X,4HTYPE / )
  OLTRTY=NSHIP(4)
  OLTRID=NSHIP(5)
  OFR=NSHIP(8)
C
C   PRINT INDIVIDUAL LIGHTER RECORD
C
C   CALL PRNIND(NSHIP,NSHORE)
C   INITIALIZE LIGHTER ID AND TYPE STATISTICS
C   CALL INID(SUMID,SUMSQID,NOID)
C   CALL INTY(SUMTY,SUMSQTY,NOTYF,NOTYR)
C   ACCUMULATE LIGHTER ID AND TYPE STATISTICS
C   CALL ACCID (SUMID,SUMSQID,NOID,NSHIP,NSHORE)
C   CALL ACCTY(SUMTY,SUMSQTY,NOTYF,NOTYR,NSHIP,NSHORE)
  OFAC=NSHIP(1)
  ODATE=NSHIP(2)
  OSHIFT=NSHIP(3)
  KPGBR= .TRUE.
  RETURN
END

```

```
SUBROUTINE PRNIND(NSHIP,NSHORE)
  DIMENSION NSHIP(11),NSHORE(11)
  WRITE(6,100)(NSHIP(J),J=4,11),NSHORE
100 FORMAT(1X, 7(R4,1X),I4,1X,2(R4,1X), 3(I4,1X,R4,1X),2(I4,1X),R4)
  RETURN
END
```

```

SUBROUTINE PRNIST( SUMID,SUMSQID,NOID)
DIMENSION SUMID(6), SUMSQID(6),XBAR(6),SD(6),NOID(3)
C COMPUTE AND PRINT INDIVIDUAL LIGHTER STATISTICS
DO 10 I=1,6
NO=NOID(1)
IF(I.EQ.2) NO=NOID(2)
IF(I.EQ.6) NO=NOID(3)
IF(NO .NE. 0) GO TO 15
XBAR(I)=0.
SD(I)=0.
GO TO 10
15 CONTINUE
XBAR(I)= SUMID(I)/NO
SD(I)= SUMSQID(I)/NO - XBAR(I)*XBAR(I)
10 CONTINUE
WRITE(6,100) NOID,XBAR,SD
100 FORMAT( 1X,11HSTATISTICS: / 5X,13H LTR MEAN (N= 3I4,2H):
1 2X , F6.1, 9X, F6.1 ,
2 4X, F6.1, 5X,F5.1,5X,2F5.1 / 7X, 8H LTR SD: 19X,F6.1,9X,F6.1,4X,
3 F6.1, 5X, F5.1, 5X, 2F5.1 )
RETURN
END

```

```

SUBROUTINE PRNTST( SUMTY,SUMSQTY,NOTYF,NOTYR )
C  COMPUTE AND PRINT LIGHTER TYPE STATISTICS
  DIMENSION SUMTY(6,2),SUMSQTY(6,2),XBAR(6,2),SD(6,2),NOTYF(3)
+      ,NOTYR(3)
  DO 10 J=1,2
    DO 20 I=1,6
      IF(J.EQ.2) GOTO 30
      N=NOTYF(1)
      IF(I.EQ.2) N=NOTYF(2)
      IF(I.EQ.6) N=NOTYF(3)
      GOTO 40
30  CONTINUE
      N=NOTYR(1)
      IF(I.EQ.2) N=NOTYR(2)
      IF(I.EQ.6) N=NOTYR(3)
40  CONTINUE
      IF(N .NE. 0) GO TO 15
      XBAR(I,J)=0.
      SD(I,J)=0.
      GO TO 20
15  CONTINUE
      XBAR(I,J)=SUMTY(I,J)/N
      SD(I,J)= SUMSQTY(I,J)/N - XBAR(I,J)*XBAR(I,J)
20  CONTINUE
10  CONTINUE
      DO 90 I=1,6
        DO 95 J=1,2
          Z9=SD(I,J)
          SD(I,J)=SQRT(Z9)
95  CONTINUE
90  CONTINUE
      WRITE(6,100)
100 FORMAT( 3X,7HFORWARD )
      WRITE(6,110) NOTYF, ( XBAR(I,1),I=1,6), (SD(J,1),J=1,6)
110 FORMAT( 5X, 13HTYPE MEAN (N= 3I4, 2H): 2X, F6.1, 9X, F6.1,
1 4X, F6.1, 5X,F5.1,5X,2F5.1 / 7X, 8HTYPE SD: 19X, F6.1, 9X, F6.1,
2 4X,F6.1, 5X, F5.1, 5X, 2F5.1 )
      WRITE(6,120)
120 FORMAT( 3X, 10HRETROGRADE )
      WRITE(6,110) NOTYR, ( XBAR(I,2),I=1,6), (SD(J,2),J=1,6)
      RETURN
      END

```

APPENDIX C

LIGHTER AND CARGO
CYCLE TIMES
PROGRAM LISTINGS


```

L5TAG,CM60000,SPUU,T500.ATG
TASK(TN=LOTS,TA=12345,OS=ORISPM,TR=TS,PI=ATG)
COMMENT.*****CTL,STMTS ON SHPDATA4*****
ATTACH,S2K,S2K260,ID=SYS2000,MR=1.
REQUEST,LRDY1,SN=SYS2.
REQUEST,LRDP1,SN=SYS2.
REQUEST,F1,SN=SYS2.
REQUEST,R1,SN=SYS2.
S2K,CR=77.
CATALOG,LRDY1,LRDY14,ID=LOTS.
REWIND,LRDY1.
COPYBF,LRDY1.
CATALOG,LRDP1,LRDP14,ID=LOTS.
REWIND,LRDP1.
COPYBF,LRDP1.
CATALOG,F1,F14,ID=LOTS.
REWIND,F1.
COPYBF,F1.
CATALOG,R1,R14,ID=LOTS.
REWIND,R1.
COPYBF,R1.
*EOR
USER,LOTS;SHARED DBN IS JLMT4A;
REPORT FILE IS LRDY1;
LI/TITLE L(4),L(4),L(4),L(4),L(4),L(4),L(4),L(4)/
C1010,C1110,C1120,C1530,C1510,C1520,C1540,C1920,
OB C1010,C1110,C1120,C1530,C1510,C1520
WH (C1010 EQ TCDF OR C1010 EQ COD)
AND C1910 EQ LRDY;
REPORT FILE IS LRDP1;
LI/TITLE L(4),L(4),L(4),L(4),L(4),L(4),L(4),L(4)/
C1010,C1110,C1120,C1530,C1510,C1520,C1920,
OB C1010,C1110,C1120,C1530,C1510,C1520
WH (C1010 EQ TCDF OR C1010 EQ COD)
AND C1910 EQ LRDP;
REPORT FILE IS F1;
LI/TITLE L(4),L(4),L(4),L(4),L(4),L(4),L(4),L(4)/
C1010,C1110,C1120,C1530,C1510,C1520,
OB C1010,C1110,C1120,C1530,C1510,C1520
WH (C1010 EQ TCDF OR C1010 EQ COD)
AND C1820 EXISTS;
REPORT FILE IS R1;
LI/TITLE L(4),L(4),L(4),L(4),L(4),L(4),L(4),L(4)/
C1010,C1110,C1120,C1530,C1510,C1520,
OB C1010,C1110,C1120,C1530,C1510,C1520
WH (C1010 EQ TCDF OR C1010 EQ COD)
AND C1720 EXISTS;
EXIT;
*EOR
*EOF

```

LSTAG,CM60000,SPUU,T50,ATG
TASK(TN=LOTS,TA=12345,OS=ORISPM,TR=TS,PI=ATG)
COMMENT.*****CTL.STMTS ON STEPS 5 & 6*****
ATTACH,S2K,S2K260,ID=SYS2000,MR=1.
REQUEST,UNIT5,*PF.
REQUEST,UNIT6,*PF.
S2K,CR=77.
CATALOG,UNIT5,UNIT45,ID=LOTS.
REWIND,UNIT5.
COPYSBF,UNIT5.
CATALOG,UNIT6,UNIT46,ID=LOTS.
REWIND,UNIT6.
COPYSBF,UNIT6.
*EOR
USER,LOTS;SHARED DBN IS JLMT4A;
REPORT FILE IS UNIT5;
LI/TITLE L(4),L(4),L(4),L(4),L(4),L(4),L(4),L(4),L(4)/
C1010,C1110,C1120,C1530,C1510,C1520,C1540,C1810,C1820,
OB C1010,C1110,C1120,C1530,C1510,C1520,C1820
WH (C1010 EQ COD OR C1010 EQ TCDF) AND C1820 EXISTS;
REPORT FILE IS UNIT6;
LI/TITLE L(4),L(4),L(4),L(4),L(4),L(4),L(4),L(4),L(4)/
C1010,C1110,C1120,C1530,C1510,C1520,C1540,C1710,C1720,
OB C1010,C1110,C1120,C1530,C1510,C1520,C1720
WH (C1010 EQ COD OR C1010 EQ TCDF) AND C1720 EXISTS;
EXIT;
*EOR
*EOF

L5TAG,CM60000,SPUU,T150.ATG
TASK(TN=LOTS,TA=12345,OS=ORISPM,TR=TS,PI=ATG)
COMMENT.*****CTL.STMTS ON STEP9*****
ATTACH,S2K,S2K260,ID=SYS2000,MR=1.
REQUEST,UNIT9A,*PF.
REQUEST,UNIT9B,*PF.
S2K,CR=77.
CATALOG,UNIT9A,UNIT49A,ID=LOTS.
REWIND,UNIT9A.
COPYSBF,UNIT9A.
CATALOG,UNIT9B,UNIT49B,ID=LOTS.
REWIND,UNIT9B.
COPYSBF,UNIT9B.
*EOR
USER,LOTS;SHARED DBN IS JLMT4A;
REPORT FILE IS UNIT9A;
LI/TITLE L(4),L(4),L(4),L(4),L(4),L(4),L(4),L(4),L(4),L(4),L(4),
L(4),L(4)/
C1010,C1110,C1120,C1365,C1330,C1335,C1305,
C1310,C1410,C1420,C1430,C1440,C1450,
OB C1010,C1110,C1120,C1330,C1335,C1305,C1310
WH (C1010 EQ TCDF OR C1010 EQ COD) AND C1330 NE NSSC;
REPORT FILE IS UNIT9B;
LI/TITLE L(4),L(4),L(4),L(4),L(4),L(4),L(4),L(4),L(4),L(4),L(4),
L(4),L(4)/
C1010,C1110,C1120,C1365,C1320,C1325,C1305,
C1310,C1410,C1420,C1430,C1440,C1450,
OB C1010,C1110,C1120,C1320,C1325,C1305,C1310
WH (C1010 EQ TCDF OR C1010 EQ COD) AND C1320 NE NSSC;
EXIT;
*EOR
*EOF

```

LSTAG,CM65000,SPUU,T400,ATG
TASK(TN=LOTS,TA=12345,OS=ORISPM,TR=TS,PI=ATG)
ATTACH,F,F14,ID=LOTS.
ATTACH,R,R14,ID=LOTS.
REWIND,F.
REWIND,R.
FILE(F,MRL=80,RT=Z,BT=C,FL=80)
FILE(R,MRL=80,RT=Z,BT=C,FL=80)
FILE(FR,MRL=80,RT=Z,BT=C,FL=80)
SORTMRG.
ATTACH,SHPGEN,SHPGENOBJ,ID=LOTS.
ATTACH,LRDY1,LRDY14,ID=LOTS.
ATTACH,LRDP1,LRDP14,ID=LOTS.
SHPGEN(,FR).
REWIND,OUTC1.
COPYSBF,OUTC1.
REWIND,OUTI1.
COPYSBF,OUTI1.
REWIND,OUTBN1.
FILE(OUTBN1,MRL=120)
FILE(UNIT2)
SORTMRG.
ATTACH,PREV,PREVOBJ,ID=LOTS.
PREV(UNIT2,UNIT3,)
REWIND,UNIT3.
FILE(UNIT3,MRL=130)
FILE(UNIT4)
SORTMRG.
ATTACH,FOR,UNIT45,ID=LOTS.
ATTACH,RET,UNIT46,ID=LOTS.
REWIND,FOR.
REWIND,RET.
FILE(FOR,MRL=80,RT=Z,BT=C,FL=80)
FILE(RET,MRL=80,RT=Z,BT=C,FL=80)
FILE(UNIT7,MRL=80,RT=Z,BT=C,FL=80)
SORTMRG.
ATTACH,STEP8,STEP8OBJ,ID=LOTS)
STEP8.
REWIND,OUTC7.
COPYSBF,OUTC7.
REWIND,OUTI7.
COPYSBF,OUTI7.
REWIND,UNIT8.
FILE(UNIT8,MRL=150)
FILE(UNIT8A)
SORTMRG.
ATTACH,FILE1,UNIT49A,ID=LOTS.
ATTACH,FILE2,UNIT49B,ID=LOTS.
FILE(FILE1,MRL=100,RT=Z,BT=C,FL=100)
FILE(FILE2,MRL=100,RT=Z,BT=C,FL=100)

```

```

FILE(UNIT10,MRL=100,RT=Z,BT=C,FL=100)
SORTMRG.
ATTACH,STEP11,STEP11OBJ,ID=LOTS.
STEP11(UNIT10,OUTI11,OUTC11,,)
REWIND,OUTC11.
COPYSBF,OUTC11.
REWIND,OUTI11.
COPYSBF,OUTI11.
REWIND,UNIT11.
FILE(UNIT11,MRL=180)
FILE(UNIT12)
SORTMRG.
ATTACH,STEP13,STEP13OBJ,ID=LOTS.
STEP13(UNIT12,UNIT13,,)
REWIND,UNIT13.
FILE(UNIT13,MRL=230)
FILE(UNIT14)
SORTMRG.
ATTACH,STEP15,STEP15OBJ,ID=LOTS.
STEP15.
REWIND,OUTC15.
COPYSBF,OUTC15.
REWIND,OUTI15.
COPYSBF,OUTI15.
REWIND,UNIT15.
FILE(UNIT15,MRL=320)
FILE(UNIT16)
SORTMRG.
ATTACH,STEP17,STEP17OBJ,ID=LOTS.
STEP17(UNIT16,,)
*EOR
MERGE
FILE,MERGE=F,R,OUTPUT=FR
FIELD,FACID(4,4,DISPLAY),DATE(11,4,DISPLAY),SHIFT(18,4,DISPLAY),
,LITYP(25,4,DISPLAY),LITID(32,4,DISPLAY),LITCY(39,4,DISPLAY),
,FR(46,4,DISPLAY)
KEY,FACID(A,DISPLAY),DATE(A,DISPLAY),SHIFT(A,DISPLAY),
,LITYP(A,DISPLAY),LITID(A,DISPLAY),LITCY(A,DISPLAY)
END
*EOR
SORT
FILE,SORT=OUTBN1,OUTPUT=UNIT2
FIELD,FACID(1,10,DISPLAY),DATE(11,10,DISPLAY),SHIFT(21,10,DISPLAY),
,LITYP(31,10,DISPLAY),LITID(41,10,DISPLAY),LITCY(51,10,DISPLAY),
,LITPS(61,10,DISPLAY),DIR(71,10,DISPLAY),LRDY(81,10,DISPLAY),
,LRDP(91,10,DISPLAY),LSPT(101,10,DISPLAY),TIME(111,10,DISPLAY)
KEY,FACID(A,DISPLAY),TIME(A,DISPLAY)
END
*EOR
SORT

```

```

FILE, SORT=UNIT3, OUTPUT=UNIT4
FIELD, FACID(1,10,DISPLAY), DATE(11,10,DISPLAY), SHIFT(21,10,DISPLAY),
, LITTY(31,10,DISPLAY), LITID(41,10,DISPLAY), LITCY(51,10,DISPLAY),
, LITPS(61,10,DISPLAY), DIR(71,10,DISPLAY), LRDY(81,10,DISPLAY),
, LRDP(91,10,DISPLAY), LSPT(101,10,DISPLAY), SUCC(111,10,DISPLAY),
, LITSP(121,10,DISPLAY)
KEY, FACID(A,DISPLAY), DATE(A,DISPLAY), SHIFT(A,DISPLAY),
, LITTY(A,DISPLAY), LITID(A,DISPLAY), LITCY(A,DISPLAY)
END
*EOR
MERGE
FILE, MERGE=FOR, RET, OUTPUT=UNIT7
FIELD, FACID(4,4,DISPLAY), DATE(11,4,DISPLAY), SHIFT(18,4,DISPLAY),
, LITTY(25,4,DISPLAY), LITID(32,4,DISPLAY), LITCY(39,4,DISPLAY),
, LITPS(46,4,DISPLAY), CARTY(53,4,DISPLAY), CARID(60,4,DISPLAY),
KEY, FACID(A,DISPLAY), DATE(A,DISPLAY), SHIFT(A,DISPLAY),
, LITTY(A,DISPLAY), LITID(A,DISPLAY), LITCY(A,DISPLAY), CARID(A,DISPLAY)
END
*EOR
SORT
FILE, SORT=UNIT8, OUTPUT=UNIT8A
FIELD, FACID(1,10,DISPLAY), DATE(11,10,DISPLAY), SHIFT(21,10,DISPLAY),
, LITTY(31,10,DISPLAY), LITID(41,10,DISPLAY), LITCY(51,10,DISPLAY),
, LITPS(61,10,DISPLAY), DIR(71,10,DISPLAY), LRDY(81,10,DISPLAY),
, LRDP(91,10,DISPLAY), LSPT(101,10,DISPLAY), SUCC(111,10,DISPLAY),
, LITSP(121,10,DISPLAY), TYPE(131,10,DISPLAY), CARGO(141,10,DISPLAY)
KEY, FACID(A,DISPLAY),
, LITTY(A,DISPLAY), LITID(A,DISPLAY), CARGO(A,DISPLAY),
, DATE(A,DISPLAY), SHIFT(A,DISPLAY)
END
*EOR
MERGE
FILE, MERGE=FILE1, FILE2, OUTPUT=UNIT10
FIELD, FACID(4,4,DISPLAY), DATE(11,4,DISPLAY), SHIFT(18,4,DISPLAY),
, OPR(25,4,DISPLAY), CARTY(32,4,DISPLAY), CARID(39,4,DISPLAY),
, CR60ID(46,4,DISPLAY), CRG0ID(53,4,DISPLAY),
, NAME(60,4,DISPLAY), TIME(67,4,DISPLAY), DELAY(74,4,DISPLAY),
, CAT(81,4,DISPLAY), DUR(88,4,DISPLAY)
KEY, FACID(A,DISPLAY), DATE(A,DISPLAY), SHIFT(A,DISPLAY),
, CARTY(A,DISPLAY), CARID(A,DISPLAY)
END
*EOR
SORT
FILE, SORT=UNIT11, OUTPUT=UNIT12
FIELD, FACID(1,10,DISPLAY), DATE(11,10,DISPLAY), SHIFT(21,10,DISPLAY),
, OPER(31,10,DISPLAY), LITTY(41,10,DISPLAY), LITID(51,10,DISPLAY),
, CARID(61,10,DISPLAY), CYCLE(71,10,DISPLAY), LDOC(81,10,DISPLAY),
, LDLK(91,10,DISPLAY), LIFT(101,10,DISPLAY), CPSN(111,10,DISPLAY),
, LAND(121,10,DISPLAY), LDUN(131,10,DISPLAY), DELY(141,10,DISPLAY),
, TYPE(151,10,DISPLAY), SEQ(161,10,DISPLAY), TIME(171,10,DISPLAY)

```

```

KEY,FACID(A,DISPLAY),TIME(A,DISPLAY),SEQ(A,DISPLAY)
END
*ECR
SORT
FILE, SORT=UNIT13, OUTPUT=UNIT14
FIELD, FACID(1,10,DISPLAY), DATE(11,10,DISPLAY), SHIFT(21,10,DISPLAY),
, OPID(31,10,DISPLAY), LITTY(41,10,DISPLAY), LITID(51,10,DISPLAY),
, CARGO(61,10,DISPLAY), CY(71,10,DISPLAY), LDOC(81,10,DISPLAY),
, DIFF1(91,10,DISPLAY), LDLK(101,10,DISPLAY), DIFF2(111,10,DISPLAY),
, LIFT(121,10,DISPLAY), DIFF3(131,10,DISPLAY), CPSN(141,10,DISPLAY),
, DIFF4(151,10,DISPLAY), LAND(161,10,DISPLAY), DIFF5(171,10,DISPLAY),
, LDUN(181,10,DISPLAY), DIFF6(191,10,DISPLAY), DELY(201,10,DISPLAY),
, TYPE(211,10,DISPLAY), SEQ(221,10,DISPLAY)
KEY, FACID(A,DISPLAY),
, LITTY(A,DISPLAY), LITID(A,DISPLAY), CARGO(A,DISPLAY),
, SEQ(A,DISPLAY), DATE(A,DISPLAY), SHIFT(A,DISPLAY)
END
*EOR
SORT
FILE, SORT=UNIT15, OUTPUT=UNIT16
FIELD, FACID(1,10,DISPLAY), DATE(11,10,DISPLAY), SHIFT(21,10,DISPLAY),
, LITTY(31,10,DISPLAY), LITID(41,10,DISPLAY), LITCY(51,10,DISPLAY),
, LITPS(61,10,DISPLAY), DIR(71,10,DISPLAY), LRDY(81,10,DISPLAY),
, LRDP(91,10,DISPLAY), LSPT(101,10,DISPLAY), SUCC(111,10,DISPLAY),
, LITSP(121,10,DISPLAY), TYPE(131,10,DISPLAY), CARGO(141,10,DISPLAY),
, OPID(151,10,DISPLAY), LDOC(161,10,DISPLAY),
, DIFF1(171,10,DISPLAY), LDLK(181,10,DISPLAY), DIFF2(191,10,DISPLAY),
, LIFT(201,10,DISPLAY), DIFF3(211,10,DISPLAY), CPSN(221,10,DISPLAY),
, DIFF4(231,10,DISPLAY), LAND(241,10,DISPLAY), DIFF5(251,10,DISPLAY),
, LDUN(261,10,DISPLAY), DIFF6(271,10,DISPLAY), DELY(281,10,DISPLAY),
, TYPE1(291,10,DISPLAY), SEQ(301,10,DISPLAY), TIME(311,10,DISPLAY)
KEY, FACID(A,DISPLAY), DATE(A,DISPLAY), SHIFT(A,DISPLAY),
, LITTY(A,DISPLAY), LITID(A,DISPLAY),
, DIR(A,DISPLAY), TIME(A,DISPLAY), SEQ(A,DISPLAY)
END
*EOR
*EOF

```

```

PROGRAM SHPGEN(LRDY1,LRDP1,FR1,OUTI1,OUTC1,OUTPUT,OUTEN1,
+           TAPE1=LRDY1,TAPE2=LRDP1,TAPE3=FR1,
+           TAPE4=OUTI1,TAPE5=OUTC1,TAPE6=OUTPUT,TAPE7=OUTEN1)

C
C   LOGICAL NFIRST,NFINC,NFCOM,NSTOP,NSTOP1,MATCH
C
C   DIMENSION KODE(3),NLRDY(6),NLRDP(6),NFR(6),NPAM(3)
C   DIMENSION NEND(3),NEOF(10)
C
C   COMMON/TBLUNIT/NUNIT1,NUNIT2,NUNIT3,NUNIT4,NUNIT5,NUNIT6,NUNIT7
C
C   INITIALIZE
C   NUNIT1=1
C   NUNIT2=2
C   NUNIT3=3
C   NUNIT4=4
C   NUNIT5=5
C   NUNIT6=6
C   NUNIT7=7
C
C   NFIRST=.FALSE.
C   NFINC=.FALSE.
C   NFCOM=.FALSE.
C   NSTOP=.FALSE.
C   NSTOP1=.FALSE.
C
C   10 CONTINUE
C   SET KODE TO READ THREE FILES.
C   KODE(1)=1
C   KODE(2)=2
C   KODE(3)=3
C
C   READ THREE RECORDS.
C   CALL READ(KODE,NLRDY,NLRDP,NFR,NLGTPS,NPAM,NFIRST,NEND)
C
C   CHECK FOR EOF IN ANY FILE
C   CALL STOP(NEND,NSTOP,NSTOP1,KODE)
C   IF(NSTOP) GO TO 60
C
C   30 CONTINUE
C   COMPARE LRDY AND LRDP RECORDS.
C   CALL COMPARE(NLRDY,NLRDP,MATCH,NFAIL1,NFAIL2)
C
C   CHECK FOR A MATCH
C   IF(MATCH) GO TO 40
C
C   THE MATCH FAILED.  SELECT WHICH FILE TO READ.
C   CALL SELECT(1,NFAIL1,NFAIL2,KODE)
C

```



```

C      WRITE THE RECORD THAT DID NOT HAVE A MATCH.
C      CALL RPTINC(KODE,NLRDY,NLRDP,NFR,NLGTPS,NPAM,NFINC)
C
C      READ A NEW RECORD.
C      CALL READ (KODE,NLRDY,NLRDP,NFR,NLGTPS,NPAM,NFIRST,NEND)
C
C      CHECK FOR EOF IN ANY FILE.
C      CALL STOP(NEND,NSTOP,NSTOP1,KODE)
C      IF(NSTOP) GO TO 60
C
C      START THE LRDY,LRDP COMPARISON OVER.
C      GO TO 30
C
C      40 CONTINUE
C      A MATCH WAS FOUND ON THE LRDY AND LRDP RECORDS.
C      COMPARE THE LRDY AND THE F/R RECORDS.
C      CALL COMPARE(NLRDY,NFR,MATCH,NFAIL1,NFAIL2)
C
C      CHECK FOR A MATCH.
C      IF(MATCH) GO TO 50
C
C      THE MATCH FAILED.  SELECT WHICH FILE TO READ.
C      CALL SELECT(2,NFAIL1,NFAIL2,KODE)
C
C      WRITE THE RECORD THAT DID NOT HAVE A MATCH.
C      CALL RPTINC(KODE,NLRDY,NLRDP,NFR,NLGTPS,NPAM,NFINC)
C
C      READ A NEW RECORD.
C      CALL READ(KODE,NLRDY,NLRDP,NFR,NLGTPS,NPAM,NFIRST,NEND)
C
C      CHECK FOR EOF IN ANY FILE
C      CALL STOP(NEND,NSTOP,NSTOP1,KODE)
C      IF(NSTOP) GO TO 60
C
C      IF LRDY AND LRDP RECORDS WERE READ THAN GO TO LRDY,
C      LRDP COMPARE.
C      IF(KODE(1).EQ.1.AND.KODE(2).EQ.2) GO TO 30
C
C      ELSE GO TO LRDY,FR COMPARE
C      GO TO 40
C
C      50 CONTINUE
C      A MATCH WAS FOUND ON THE LRDY,LRDP, AND FR RECORDS.
C      CONVERT LRDY TO MINS FROM THE BEGINNING OF THE TEST.
C      CALL CONVERT(NLRDY(2),NLRDY(3),NPAM(1),MINS)
C
C      CONVERT LRDP TO MINS FROM THE BEGINNING OF THE TEST.
C      CALL CONVERT(NLRDY(2),NLRDY(3),NPAM(2),MINS1)
C

```

```

C      CALCULATE LIGHTER AT SHIP TIME
      LSHPT=MINS1-MINS
C
C      WRITE A COMPLETE RECORD.
      CALL RPTCOM(NLRDY,NLGTPS,NPAM,NFCOM,LSHPT,MINS)
C
C      START OVER WITH READING THREE NEW RECORDS.
      GO TO 10
C
C
60 CONTINUE
C      CHECK FOR EOF IN THREE FILES.
      IF(NSTOP1) GO TO 70
C
C      WRITE INCOMPLETE RECORDS.
      CALL RPTINC(KODE,NLRDY,NLRDP,NFR,NLGTPS,NPAM,NFINC)
C      READ A NEW RECORD(S)
      CALL READ(KODE,NLRDY,NLRDP,NFR,NLGTPS,NPAM,NFIRST,NEND)
C
C      CHECK FOR END CONDITION.
      CALL STOP(NEND,NSTOP,NSTOP1,KODE)
C
C      CONTINUE STOP PROCESS
      GO TO 60
C
C
C
70 CONTINUE
C
C      WRITE EOF IN BINARY FILE
      DO 80 J=1,10
        NEOF(J)=4RZEOF
80 CONTINUE
      WRITE(NUNIT7) NEOF,LSHPT,MINS
C
      END

```

```

C      SUBROUTINE READ(KODE,NLRDY,NLRDP,NFR,NLGTPS,NPAM,NFIRST,NEND)
C
C      PURPOSE: READS ONE RECORD FROM ANY OF THREE FILES DEPENDING
C                ON THE VALUE OF KODE
C
C      LOGICAL NFIRST
C
C      DIMENSION KODE(3),NLRDY(6),NLRDP(6),NFR(6),NPAM(3),NEND(3)
C
C      COMMON/TBLUNIT/NUNIT1,NUNIT2,NUNIT3,NUNIT4,NUNIT5,NUNIT6,NUNIT7
C
C      CHECK TO DETERMINE IF THE FILES NEED REWINDING
C      IF(NFIRST) GO TO 10
C          REWIND NUNIT1
C          REWIND NUNIT2
C          REWIND NUNIT3
C          REWIND NUNIT4
C          REWIND NUNIT5
C          REWIND NUNIT7
C          NFIRST=.TRUE.
C
C      10 CONTINUE
C
C      IF KODE EQ 1 THAN READ THE LRDY FILE
C      IF(KODE(1).NE.1) GO TO 30
C          READ(NUNIT1,20) NLRDY,NLGTPS,NPAM(1)
C      20  FORMAT(3X,8(R4,3X))
C          SAVE EOF KODE
C          NEND(1)=NLRDY(1)
C      30 CONTINUE
C
C      IF KODE EQ 2 THAN READ THE LRDP FILE
C      IF(KODE(2).NE.2) GO TO 50
C          READ(NUNIT2,40) NLRDP,NPAM(2)
C      40  FORMAT(3X,7(R4,3X))
C          SAVE EOF KODE
C          NEND(2)=NLRDP(1)
C      50 CONTINUE
C
C      IF KODE EQ 3 THEN READ THE R/F FILE
C      IF(KODE(3).NE.3) GO TO 60
C          READ(NUNIT3,40) NFR,NPAM(3)
C          SAVE EOF KODE
C          NEND(3)=NFR(1)
C      60 CONTINUE
C
C      RETURN
C      END

```

```

C      SUBROUTINE COMPARE(NTEST1,NTEST2,MATCH,NFAIL1,NFAIL2)
C
C      PURPOSE: DETERMINES IF TWO RECORDS MATCH ON SIX MATCH KEYS.
C                IF NO MATCH, RETURNS TO THE MAIN PROGRAM THE FIRST
C                MATCH KEYS WHERE THE MATCH FAILED.
C
C      LOGICAL MATCH
C
C      DIMENSION NTEST1(6),NTEST2(6)
C
C      NTEST1 HAS THE LRDY RECORD.
C      NTEST2 HAS THE LRDP OR F/R RECORD.
C      COMPARE ON THE SIX MATCH KEYS.
C      DO 10 J=1,6
C          IF(NTEST1(J).EQ.NTEST2(J)) GO TO 10
C          THE MATCH FAILS.
C          MATCH=.FALSE.
C          NFAIL1 HAS THE LRDY MATCH KEY
C          NFAIL1=NTEST1(J)
C          NFAIL2 HAS THE LRDP OR F/R MATCH KEY
C          NFAIL2=NTEST2(J)
C
C      RETURN
C
C 10 CONTINUE
C
C      A MATCH WAS FOUND
C      MATCH=.TRUE.
C
C      RETURN
C      END

```

```

C      SUBROUTINE SELECT(KEY,NFAIL1,NFAIL2,KODE)
C
C      PURPOSE: DETERMINES WHICH RECORD TO READ.
C
C      DIMENSION KODE(3)
C
C      KEY=1 WHEN MATCH FAILED ON LRDY,LRDP RECORDS
C      KEY=2 WHEN MATCH FAILED ON LRDY,NF/R RECORDS
C
C      GO TO (10,20) KEY
C
10  CONTINUE
C      THIS SECTION IS FOR KEY=1.  A NON-MATCH BETWEEN A LRDY RECORD
C      AND A LRDP RECORD.
C
C      TEST TO DETERMINE WHICH FILE (LRDY,LRDP) TO READ.
C      IF MATCH KEY LRDY GT MATCH KEY LRDP THEN READ LRDP RECORD.
C      ELSE READ LRDY RECORD
C      KODE(1)=0
C      KODE(2)=2
C      KODE(3)=0
C      IF(NFAIL1.GT.NFAIL2) GO TO 15
C          KODE(1)=1
C          KODE(2)=0
C          KODE(3)=0
15  CONTINUE
C
C      RETURN
C
20  CONTINUE
C
C      THIS SECTION IS FOR KEY=2.  A NON-MATCH BETWEEN LRDY AND NF/R
C      FILES.
C
C      TEST TO DETERMINE WHICH FILE (LRDY,NF/R TO READ.
C      IF MATCH KEY LRDY IS LT MATCH KEY R/F THAN READ LRDY
C      AND LRDP RECORDS.
C      ELSE READ R/F RECORD
C      KODE(1)=1
C      KODE(2)=2
C      KODE(3)=0
C      IF(NFAIL1.LT.NFAIL2) GO TO 30
C          KODE(1)=0
C          KODE(2)=0
C          KODE(3)=3
30  CONTINUE
C
C      RETURN
C      END

```

```

C      SUBROUTINE CONVERT(NDAY,NSHFT,NHM,MINS)
C
C      PURPOSE: CONVERTS DAYS,SHIFT,HRS AND MINS TO MINS FROM BEGINNING
C                OF THE TEST
C
C      DECODE(10,10,NDAY) ND
10  FORMAT(6X,I4)
C
C      DECODE(10,10,NSHFT) NS
C
C      DECODE(10,20,NHM) NH,NM
20  FORMAT(6X,I2,I2)
C
C
C      ADD 24 TO HRS LT 18 ON SHIFT 2
      IF(NS.EQ.2.AND.NH.LT.18) NH=NH+24
C
C
C      CALCULATE DAY OF TEST. DAY1 EQ 0.
      ND=ND-218
C
C      CALCULATE MINS.
      MINS=ND*24*60+NH*60+NM
C
C
      RETURN
      END

```



```

C      SUBROUTINE RPTCOM(NLRDY,NLGTPS,NPAM,NFCOM,LSHPT,MINS)
C
C      PURPOSE: WRITES A COMPLETE RECORD
C
C      LOGICAL NFCOM
C
C      DIMENSION NLRD:(6),NPAM(3)
C
C      COMMON/TBLUNIT/NUNIT1,NUNIT2,NUNIT3,NUNIT4,NUNIT5,NUNIT6,NUNIT7
C
C      IF FIRST THAN PRINT HEADER
C      IF(NFCOM) GO TO 20
C          XDATE=DATE(JDUMMY)
C          XTIME=TIME(JDUMMY)
C          WRITE(NUNIT5,10) XDATE,XTIME
10      FORMAT(1X,16HCOMPLETE RECORDS,2(5X,A10),3(5X,R4))
C          NFCOM=.TRUE.
20      CONTINUE
C
C      WRITE A COMPLETE RECORD
C      WRITE(NUNIT5,30) NLRDY,NLGTPS,NPAM(3),NPAM(1),NPAM(2),LSHPT,MINS
30      FORMAT(1X,10(R4,3X),I4,3X,I6)
C
C      WRITE(NUNIT7) NLRDY,NLGTPS,NPAM(3),NPAM(1),NPAM(2),LSHPT,MINS
C
C      RETURN
C      END

```



```

SUBROUTINE STOP(NEND,NSTOP,NSTOP1,KODE)
C
C PURPOSE: TO DETERMINE IF EOF IN ANY FILE. TO DETERMINE IF
C           EOF IN ALL FILES. SET KODE TO WRITE INCOMPLETE RECORDS.
C
C LOGICAL NSTOP,NSTOP1
C
C DIMENSION NEND(3), KODE(3)
C
C TEST FOR AT LEAST ONE EOF.
C IF(NSTOP) GO TO 60
C   DO 20 J=1,3
C     IF(NEND(J).EQ.4R EOF) GO TO 30
C     NO EOF FOUND
20  CONTINUE
C
C ALL FILES CONTAIN DATA
C NSTOP=.FALSE.
C
C RETURN
C
30  CONTINUE
C SET KODE FOR AT LEAST ONE EOF
C NSTOP=.TRUE.
C
60  CONTINUE
C CHECK FOR EOF IN LRDY FILE. SET KODE.
C KODE(1)=0
C IF(NEND(1).EQ.4R EOF) GO TO 80
C   KODE(1)=1
80  CONTINUE
C
C CHECK FOR EOF IN LRDP FILE. SET KODE.
C KODE(2)=0
C IF(NEND(2).EQ.4R EOF) GO TO 100
C   KODE(2)=2
100 CONTINUE
C
C CHECK FOR EOF IN THE F/R FILE. SET KODE.
C KODE(3)=0
C IF(NEND(3).EQ.4R EOF) GO TO 120
C   KODE(3)=3
120 CONTINUE
C
C CHECK FOR EOF IN ALL FILES.
C IF(KODE(1).EQ.0.AND.KODE(2).EQ.0.AND.KODE(3).EQ.0) NSTOP1=.TRUE.
C
C RETURN
C END

```

```

C      PROGRAM PREV(TAPE1,OUT,OUTPUT,TAPE2=OUT,TAPE6=OUTPUT)
C
C      PURPOSE: TO SAVE PREVIOUS LRDP ON PRESENT LIGHTER AT SHIP RECORD,
C                AND TO COMPUTE LIGHTER SUCCESSION TIME AND LIGHTER AT
C                SHIP CYCLE TIME.
C
C      DIMENSION N(14)
C
C      REWIND 1
C      REWIND 2
C      NOLD=4RNONE
C
C      REPEAT UNTIL OUT OF DATA
5  CONTINUE
      READ(1) (N(I),I=1,12)
C
C      IF(N(1) .EQ. 4RZEOF) GO TO 10
C
C      N(13)=-999
C      N(14)=-999
C      CALL CONVERT(N(2),N(3),N(10),LRDP)
C      IF(N(1) .NE. NOLD) GO TO 10
C
C      COMPUTE LIGHTER SUCC TIME (LRDY-PREV LRDP)
C      N(13)=N(12)-NLRDP
C      COMPUTE LIGHTER AT SHIP CYCLE TIME (LRDP-PREV LRDP)
C      N(14)=LRDP-NLRDP
C
10  CONTINUE
C
C      WRITE(2) (N(I),I=1,11),N(13),N(14)
C
C      WRITE(6,15) (N(I),I=1,11),N(13),N(14)
15  FORMAT(10(3X,R4),3X,I4,3X,I6,2(3X,I4))
C
C      NLRDP=LRDP
C      NOLD=N(1)
C
C      IF(N(1) .EQ. 4RZEOF) GO TO 30
C
C      GO TO 5
30 CONTINUE
C
C      END

```

```

C      SUBROUTINE CONVERT(NDAY,NSHFT,NHM,MINS)
C
C      PURPOSE: CONVERTS DAYS,SHIFT,HRS AND MINS TO MINS FROM BEGINNING
C                OF THE TEST
C
C      DECODE(10,10,NDAY) ND
10     FORMAT(6X,I4)
C
C      DECODE(10,10,NSHFT) NS
C
C      DECODE(10,20,NHM) NH,NM
20     FORMAT(6X,I2,I2)
C
C
C      ADD 24 TO HRS LT 17 ON SHIFT 2
      IF(NS.EQ.2.AND.NH.LT.17) NH=NH+24
C
C
C      CALCULATE DAY OF TEST. DAY1 EQ 0.
      ND=ND-218
C
C      CALCULATE MINS.
      MINS=ND*24*60 + NH*60 + NM
C
C
      RETURN
      END

```

```

PROGRAM STEP8(UNIT4,UNIT7,OUTI7,OUTC7,UNIT8,OUTPUT,
+           TAPE1=UNIT4,TAPE2=UNIT7,TAPE3=OUTI7,
+           TAPE4=OUTC7,TAPE5=UNIT8,TAPE6=OUTPUT)

C
C
C   PURPOSE: ADDS CARGO TYPE AND CARGO ID TO LIGHTER RECORDS
C
C   LOGICAL NPRINT,NSTOP,NSTOP1,MATCH
C
C   DIMENSION KODE(2),LIGHTER(13),NCARGO(9),NEND(2)
C
C   COMMON/TBLUNIT/NUNIT1,NUNIT2,NUNIT3,NUNIT4,NUNIT5,NUNIT6
C
C   INITIALIZE
C   NUNIT1=1
C   NUNIT2=2
C   NUNIT3=3
C   NUNIT4=4
C   NUNIT5=5
C   NUNIT6=6
C
C   NPRINT=.FALSE.
C   NSTOP=.FALSE.
C   NSTOP1=.FALSE.
C   MATCH=.FALSE.
C
C   SET KODE TO READ BOTH FILES
C   KODE(1)=1
C   KODE(2)=2
C
10 CONTINUE
C   READ LIGHTER AND/OR CARGO RECORDS.
C   CALL READ(KODE,LIGHTER,NCARGO,NEND)
C
C   CHECK FOR EOF IN ANY FILE.
C   CALL STOP(NEND,NSTOP,NSTOP1,KODE)
C   IF(NSTOP) GOTO 30
C
C   COMPARE LIGHTER AND CARGO RECORDS.
C   CALL COMPARE(LIGHTER,NCARGO,MATCH,NFAIL1,NFAIL2)
C
C   CHECK FOR A MATCH
C   IF(MATCH) GO TO 20
C
C   THE MATCH FAILED.  SELECT WHICH FILE TO READ.
C   CALL SELECT(NFAIL1,NFAIL2,KODE)
C
C   WRITE AN INCOMPLETE RECORD, IF ANY
C   CALL RPTINC(KODE,LIGHTER,NCARGO,NPRINT)
C
C   START OVER AND READ A NEW RECORD

```

```

      GO TO 10
C
20  CONTINUE
C   A MATCH WAS FOUND.
C   WRITE A COMPLETE RECORD.
C   CALL RPTCOM(LIGHTER,NCARGO,NPRINT)
C
C   SET KODE TO READ A NEW CARGO RECORD
C   KODE(1)=0
C   KODE(2)=2
C
C   READ A NEW CARGO RECORD AT THE BEGINNING
C   OF THE PROGRAM
C   GO TO 10
C
C
30  CONTINUE
C   CHECK FOR EOF IN BOTH FILES.
C   IF(NSTOP1) GO TO 40
C
C   WRITE INCOMPLETE RECORDS
C   CALL RPTINC(KODE,LIGHTER,NCARGO,NPRINT)
C   READ NEW RECORD
C   CALL READ(KODE,LIGHTER,NCARGO,NEND)
C
C   CHECK FOR EOF
C   CALL STOP(NEND,NSTOP,NSTOP1,KODE)
C
C   CONTINUE STOP PROCESS.
C   GO TO 30
C
C
40  CONTINUE
C   WRITE EOF IN BINARY FILE
C
C   DO 50 J=11,13
C       LIGHTER(J)=999
50  CONTINUE
C
C   DO 60 J=1,10
C       LIGHTER(J)=4RZEOF
60  CONTINUE
C
C   DO 70 J=8,9
C       NCARGO(J)=4RZEOF
70  CONTINUE
C
C   WRITE(NUNIT5)  LIGHTER,NCARGO(8),NCARGO(9)
C
C   END

```

```

SUBROUTINE READ(KODE,LIGHTER,NCARGO,NEND)
C
C PURPOSE: TO READ A RECORD FROM THE LIGHTER TIMES FILE AND/OR
C          THE CARGO ID FILE
C
C LOGICAL FIRST
C
C DIMENSION KODE(2),LIGHTER(13),NCARGO(9),NEND(2)
C
C COMMON/TBLUNIT/NUNIT1,NUNIT2,NUNIT3,NUNIT4,NUNIT5,NUNIT6
C
C DATA FIRST/6H.TRUE./
C
C REWIND FILES FIRST TIME THRU
C IF(.NOT.FIRST) GO TO 10
    REWIND NUNIT1
    REWIND NUNIT2
    REWIND NUNIT3
    REWIND NUNIT4
    REWIND NUNIT5
    FIRST=.FALSE.
10 CONTINUE
C
C IF KODE(1)=1 READ LIGHTER FILE
C IF( KODE(1) .NE. 1 ) GO TO 20
    READ(NUNIT1) LIGHTER
    NEND(1)=LIGHTER(1)
20 CONTINUE
C
C IF KODE(2)=2 READ CARGO FILE
C IF( KODE(2) .NE. 2 ) GO TO 40
    READ(NUNIT2,30) NCARGO
30  FORMAT(9(3X,R4))
    NEND(2)=NCARGO(1)
40 CONTINUE
C
C RETURN
C END

```

```

C      SUBROUTINE COMPARE(NTEST1,NTEST2,MATCH,NFAIL1,NFAIL2)
C
C      PURPOSE: DETERMINES IF TWO RECORDS MATCH ON SIX MATCH KEYS.
C                IF NO MATCH, RETURNS TO THE MAIN PROGRAM THE FIRST
C                KEY WHERE THE MATCH FAILED.
C
C      LOGICAL MATCH
C
C      DIMENSION NTEST1(13),NTEST2(9)
C
C      COMPARE ON THE SIX MATCH KEYS.
C      DO 10 J=1,6
C        IF( NTEST1(J) .EQ. NTEST2(J) ) GOTO 10
C
C        THE MATCH FAILS
C        MATCH=.FALSE.
C        NFAIL1=NTEST1(J)
C        NFAIL2=NTEST2(J)
C
C      RETURN
C
C 10 CONTINUE
C
C      A MATCH WAS FOUND
C      MATCH=.TRUE.
C
C      RETURN
C      END

```

C

C

C

C

1

C

C


```

C      SUBROUTINE RPTINC(KODE,LIGHTER,NCARGO,NPRINT)
C
C      PURPOSE: WRITES AN INCOMPLETE RECORD
C
C      LOGICAL FIRST,NPRINT
C
C      DIMENSION KODE(2),LIGHTER(13),NCARGO(9)
C
C      COMMON/TBLUNIT/NUNIT1,NUNIT2,NUNIT3,NUNIT4,NUNIT5,NUNIT6
C
C      DATA FIRST/6H.TRUE./
C
C      PRINT HEADER ON FIRST CALL
C      IF(.NOT.FIRST) GO TO 20
C          XDATE=DATE(JDUMMY)
C          XTIME=TIME(JDUMMY)
C          WRITE(NUNIT3,10) XDATE,XTIME
10      FORMAT(1H1,18HINCOMPLETE RECORDS,2(5X,A10))
C          FIRST=.FALSE.
C
C      20 CONTINUE
C      DETERMINE WHICH RECORD TO OUTPUT
C      IF(KODE(1) .NE. 1) GOTO 50
C      IF(.NOT.NPRINT) GOTO 30
C          NPRINT=.FALSE.
C          RETURN
C
C      30 CONTINUE
C      NPRINT=.FALSE.
C      WRITE(NUNIT3,40) LIGHTER
C      40 FORMAT(1X,7HLIGHTER,10(3X,R4),3(3X,I4))
C
C      RETURN
C
C
C
C      50 CONTINUE
C      WRITE(NUNIT3,60) NCARGO
C      60 FORMAT(1X,5HCARGO,9(3X,R4))
C
C
C      RETURN
C      END

```

```

C      SUBROUTINE RPTCOM(LIGHTER,NCARGO,NPRINT)
C
C      PURPOSE: WRITES A COMPLETE RECORD
C
C      LOGICAL FIRST,NPRINT
C
C      DIMENSION LIGHTER(13),NCARGO(9)
C
C      COMMON/TBLUNIT/NUNIT1,NUNIT2,NUNIT3,NUNIT4,NUNIT5,NUNIT6
C
C      DATA FIRST/6H.TRUE./
C
C      WRITE HEATER ON FIRST CALL
C      IF(.NOT.FIRST) GOTO 20
C          XDATE=DATE(JDUMMY)
C          XTIME=TIME(JDUMMY)
C          WRITE(NUNIT4,10) XDATE,XTIME
10      FORMAT(1H1,16HCOMPLETE RECORDS,2(5X,A10))
C          FIRST=.FALSE.
C
C
C      20 CONTINUE
C          WRITE A COMPLETE RECORD
C          WRITE(NUNIT4,30) LIGHTER,NCARGO(8),NCARGO(9)
30      FORMAT(1X,10(3X,R4),3(3X,I4),2(3X,R4))
C
C          WRITE(NUNIT5) LIGHTER,NCARGO(8),NCARGO(9)
C
C          SET CODE TO TRUE.  THE PRESENT LIGHTER RECORD WAS COMPLETE.
C          NPRINT=.TRUE.
C
C
C      RETURN
C      END

```

```

SUBROUTINE STOP(NEND,NSTOP,NSTOP1,KODE)
C
C PURPOSE: TO DETERMINE IF EOF IN ANY FILE. TO DETERMINE IF
C           EOF IN ALL FILES. SETS KODE TO INDICATE RECORDS.
C
C LOGICAL NSTOP,NSTOP1
C
C DIMENSION NEND(2),KODE(2)
C
C TEST FOR AT LEAST ONE EOF
C IF(NSTOP) GOTO 30
C   DO 10 J=1,2
C     IF(NEND(J).EQ.4RZEOF) GOTO 20
C   NO EOF FOUND
10 CONTINUE
C
C   ALL FILES CONTAIN DATA
C   NSTOP=.FALSE.
C
C   RETURN
C
C 20 CONTINUE
C   SET KODE FOR AT LEAST ONE EOF
C   NSTOP=.TRUE.
C
C 30 CONTINUE
C   CHECK FOR EOF IN LIGHTER FILE
C   KODE(1)=0
C   IF(NEND(1).NE.4RZEOF) KODE(1)=1
C
C   CHECK FOR EOF IN CARGO FILE
C   KODE(2)=0
C   IF(NEND(2).NE.4RZEOF) KODE(2)=2
C
C   CHECK FOR EOF IN ALL FILES.
C   IF(KODE(1).EQ.0.AND.KODE(2).EQ.0) NSTOP1=.TRUE.
C
C   RETURN
C   END

```

```

PROGRAM CARGOID(CARGO,OUTI1,OUTC1,UNIT11,OUTPUT,
+           TAPE1=CARGO,TAPE2=OUTI1,TAPE3=OUTC1,
+           TAPE4=UNIT11,TAPE6=OUTPUT)

C
C   LOGICAL STOP,FIRST,END,FLAG,FIRST1
C
C   DIMENSION NDATA(13),NPAM(6),NDELAY1(20),NDELAY2(20),
+           NSAVE(8)
C
C   COMMON/TBLUNIT/NUNIT1,NUNIT2,NUNIT3,NUNIT4
C
C   INITIALIZE
C   NUNIT1=1
C   NUNIT2=2
C   NUNIT3=3
C   NUNIT4=4
C   STOP=.FALSE.
C
10  CONTINUE
C   SET FLAG TO INITIALIZE SUBROUTINE CHANGE
C   FIRST=.TRUE.
C   SET FLAG TO INITIALIZE SUBROUTINE BUILD
C   FIRST1=.TRUE.
C
C
20  CONTINUE
C   READ A CARGO RECORD
C   CALL READ(NDATA,STOP)
C   IF(STOP) GO TO 30
C
C   CHECK FOR A CHANGE ON FACILITY ID, LIGHTER TYPE, LIGHTER ID,
C   CARGO ID OR CARGO CYCLE CHANGE.
C   CALL CHANGE(NDATA,FIRST,END)
C
C
C   TEST FOR CHANGE
C   IF(END) GO TO 30
C
C   NO CHANGE. CONTINUE TO BUILD A NPAM RECORD.
C   CALL BUILD(NDATA,FIRST1,NPAM,NDELAY1,NDELAY2,
+           NSEQ,FLAG,NSAVE)
C
C   GO TO 20
C
C
30  CONTINUE
C   TEST FOR COMPLETE OR INCOMPLETE RECORD.
C   IF(FLAG) GO TO 40
C   WRITE AN INCOMPLETE RECORD.

```

```

      CALL RPTINC(NSAVE, NPAM, NSEQ, NDELAY1, NDELAY2)
      GO TO 50
C
C
40  CONTINUE
C    COMPUTE TIME FROM 0 BASED ON LDUN
      CALL CONVERT(NSAVE(2), NSAVE(3), NPAM(6), NTIME)
C
C    WRITE A COMPLETE RECORD.
      CALL RPTCOM(NSAVE, NPAM, NSEQ, NDELAY1, NDELAY2, NTIME)
C
C
50  CONTINUE
C    CHECK FOR EOF
      IF (STOP) GO TO 60
      BACKSPACE 1
C    GO TO TOP OF PROGRAM
      GO TO 10
C
C
60  CONTINUE
      END

```

```

SUBROUTINE READ(NDATA,STOP)
C
C   PURPOSE: TO READ ONE INPUT RECORD.
C
C   LOGICAL FIRST,STOP
C   DIMENSION NDATA(13)
C
C   COMMON/TBLUNIT/NUNIT1,NUNIT2,NUNIT3,NUNIT4
C
C   DATA FIRST/6H.TRUE./
C
C   REWIND FILES THE FIRST TIME THRU
C   IF(.NOT.FIRST) GO TO 10
C       REWIND NUNIT1
C       REWIND NUNIT2
C       REWIND NUNIT3
C       REWIND NUNIT4
C       FIRST=.FALSE.
10  CONTINUE
C
C   READ ONE RECORD
C   READ(NUNIT1,20) NDATA
20  FORMAT(13(3X,R4))
C
C   CHECK FOR EOF
C   IF(EOF(NUNIT1)) 30,40
30  CONTINUE
C       STOP=.TRUE.
C
C
C   40 CONTINUE
C
C   RETURN
C   END

```

SUBROUTINE CHANGE(NDATA,FIRST,END)

PURPOSE: TO SET THE VALUE OF END TO TRUE WHEN FACILITY ID,
LIGHTER TYPE, LIGHTER ID, CARGO ID OR CARGO CYCLE CHANGE.

LOGICAL END,FIRST

DIMENSION NDATA(13)

IF(,NOT,FIRST) GO TO 10

NFACID=NDATA(1)

NLITTY=NDATA(5)

NLITID=NDATA(6)

NCARID=NDATA(7)

NCARCY=NDATA(8)

FIRST=,FALSE.

END=,FALSE.

RETURN

10 CONTINUE

SET END TO TRUE IF ANY KEY CHANGES

IF(NDATA(1) ,NE. NFACID) END=,TRUE.

IF(NDATA(5) ,NE. NLITTY) END=,TRUE.

IF(NDATA(6) ,NE. NLITID) END=,TRUE.

IF(NDATA(7) ,NE. NCARID) END=,TRUE.

IF(NDATA(8) ,NE. NCARCY) END=,TRUE.

SAVE PRESENT KEY VALUES.

NFACID=NDATA(1)

NLITTY=NDATA(5)

NLITID=NDATA(6)

NCARID=NDATA(7)

NCARCY=NDATA(8)

RETURN

END

```

SUBROUTINE BUILD(NDATA,FIRST,NPAM,NDELAY1,NDELAY2,
+               NSEQ,FLAG,NSAVE)
C   PURPOSE: TO CONSTRUCT A NPAM RECORD WHICH CONTAINS LDOC, LDLK,
C             LIFT,CPSN,LAND,LDUN AND TO CONSTRUCT A NDELAY RECORD(S)
C             WHICH CONTAINS DELAY TYPE, DELAY CATEGORY AND DELAY
C             DURATION. SETS A FLAG TO INDICATE IF THE RECORD IS
C             COMPLETE.
C
C   LOGICAL FIRST,FLAG
C
C   DIMENSION NDATA(13),NPAM(6),NAME(6),NDELAY1(20),NDELAY2(20),
+             NSAVE(8)
C   DATA NAME/4RLDOC,4RLDLK,4RLIFT,4RCPSN,4RLAND,4RLDUN/
C   IF(.NOT.FIRST) GO TO 20
C     DO 10 J=1,6
C       NPAM(J)=4RNONE
10    CONTINUE
C     NSEQ=0
C     NDELAY1(1)=4RNONE
C     NDELAY2(1)=4RNONE
C     FIRST=.FALSE.
C   SAVE FIRST EIGHT VALUES OF PRESENT NDATA
C     DO 15 J=1,8
C       NSAVE(J)=NDATA(J)
15    CONTINUE
20    CONTINUE
C
C   SET THE EVENT TIME IN THE CORRESPONDING EVENT NAME PARAMETER
C     DO 30 J=1,6
C       IF(NDATA(9) .EQ. NAME(J)) NPAM(J)=NDATA(10)
30    CONTINUE
C
C   CHECK FOR A COMPLETE NPAM RECORD
C     FLAG=.TRUE.
C     DO 40 J=1,6
C       IF(NPAM(J) .EQ. 4RNONE) FLAG=.FALSE.
40    CONTINUE
C
C   CHECK TO DETERMINE IF DELAY PARAMETERS EXIST IN NDATA RECORD.
C     IF(NDATA(11) .EQ. 4RU ) RETURN
C
C     A DELAY EXISTS
C     NSEQ=NSEQ+1
C     NDELAY1(NSEQ)=NDATA(10)
C     NDELAY2(NSEQ)=NDATA(11)
C
C   RETURN
C   END

```


SUBROUTINE CONVERT(NDAY,NSHFT,NHM,MINS)

PURPOSE: CONVERTS DAYS,SHIFT,HRS AND MINS TO MINS FROM BEGINNING
OF THE TEST

DECODE(10,10,NDAY) ND
10 FORMAT(6X,I4)

DECODE(10,10,NSHFT) NS

DECODE(10,20,NHM) NH,NM
20 FORMAT(6X,I2,I2)

ADD 24 TO HRS LT 17 ON SHIFT 2
IF(NS.EQ.2.AND.NH.LT.17) NH=NH+24

CALCULATE DAY OF TEST. DAY1 EQ 0.
ND=ND-218

CALCULATE MINS.
MINS=ND*24*60 + NH*60 + NM

RETURN
END

```

C      SUBROUTINE RPTINC(NDATA,NPAM,NSEQ,NDELAY1,NDELAY2)
C
C      PURPOSE: WRITES AN INCOMPLETE RECORD.
C
C      LOGICAL FIRST
C
C      DIMENSION NDATA(8),NPAM(6),NDELAY1(20),NDELAY2(20)
C
C      COMMON/TBLUNIT/ NUNIT1,NUNIT2,NUNIT3,NUNIT4
C
C      DATA FIRST/6H.TRUE./
C
C      WRITE HEADER ON FIRST CALL
C      IF(.NOT.FIRST) GO TO 20
C          XDATE=DATE(JDUMMY)
C          XTIME=TIME(JDUMMY)
C          WRITE(NUNIT2,10) XDATE,XTIME
10      FORMAT(1H1,24HINCOMPLETE CARGO RECORDS,2(5X,A10))
C          WRITE(NUNIT2,15)
15      FORMAT(59X,4HLD0C,3X,4HLDLK,3X,4HLIFT,3X,4HCPSN,3X,
C          +      4HLAND,3X,4HLDUN)
C          FIRST=.FALSE.
20      CONTINUE
C
C
C      WRITE INCOMPLETE CARGO RECORD
C      DO 40 J=1,NSEQ
C          WRITE(NUNIT2,30) (NDATA(I),I=1,8),NPAM,NDELAY1(J),NDELAY2(J)
30      FORMAT(16(3X,R4))
40      CONTINUE
C
C
C      RETURN
C      END

```

```

C      SUBROUTINE RPTCOM(NDATA,NPAM,NSEQ,NDELAY1,NDELAY2,NTIME)
C
C      PURPOSE: WRITES A COMPLETE RECORD.
C
C      LOGICAL FIRST
C
C      DIMENSION NDATA(8),NPAM(6),NDELAY1(20),NDELAY2(20)
C
C      COMMON/TBLUNIT/NUNIT1,NUNIT2,NUNIT3,NUNIT4
C      DATA FIRST/6H.TRUE./
C
C      WRITE HEADER ON FIRST CALL
C      IF(.NOT.FIRST) GO TO 20
C          XDATE=DATE(JDUMMY)
C          XTIME=TIME(JDUMMY)
C          WRITE(NUNIT3,10) XDATE,XTIME
10      FORMAT(1H1,22HCOMPLETE CARGO RECORDS,2(5X,A10))
C          FIRST=.FALSE.
20      CONTINUE
C
C      WRITE A COMPLETE CARGO RECORD
C      DO 40 J=1,NSEQ
C          WRITE(NUNIT3,30) (NDATA(I),I=1,8),NPAM,NDELAY1(J),
C          +                NDELAY2(J),J,NTIME
30      FORMAT(16(2X,R4),2X,I4,2X,I6)
C
C          WRITE(NUNIT4) (NDATA(I),I=1,8),NPAM,NDELAY1(J),
C          +                NDELAY2(J),J,NTIME
C
C      40 CONTINUE
C
C      RETURN
C      END

```

```

C      PROGRAM STEP13(TAPE1,TAPE2,TAPE3,OUTPUT,TAPE4=OUTPUT)
C
C      PURPOSE: TO WRITE PREVIOUS LDUN ON PRESENT RECORD AND TO
C                COMPUTE THE PARTS OF THE CRANE CYCLE.
C
C      DIMENSION N(18),NPREV(3),J(6)
C
C      REWIND 1
C      REWIND 2
C      REWIND 3
C
C      REPEAT UNTIL OUT OF DATA
10  CONTINUE
    READ(1) N
C
C      IF(EOF(1)) 30,20
20      CONTINUE
C
C      OBTAIN PREVIOUS DATE,SHIFT AND LDUN.
C      CALL SAVE(N(1),N(2),N(3),N(14),NPREV)
C
C      COMPUTE CPMPOINT PARTS OF A CRANE CYCLE.
C      CALL COMPUTE(N,NPREV,J)
C
C      WRITE RESULTS
C      WRITE(2) ((N(I),I=1,9),J(1),N(10),J(2),N(11),J(3),N(12),
+              J(4),N(13),J(5),N(14),J(6),(N(I),I=15,17))
C
C      WRITE(3,25) ((N(I),I=1,9),J(1),N(10),J(2),N(11),J(3),N(12),
+              J(4),N(13),J(5),N(14),J(6),(N(I),I=15,17))
C
C      25  FORMAT(1X,9(R4,1X),6(I4,1X,R4,1X),R4,1X,I4)
C
C      GO TO 10
C
C      30 CONTINUE
C
C      END

```

```

SUBROUTINE COMPUTE(NDATA,NPREV,NTIME)
C
C PURPOSE: TO COMPUTE COMPONT PARTS OF A CRANE CYCLE.
C
C DIMENSION NDATA(17),NPREV(3),NTIME(6)
C
C CONVERT ELAPSED CYCLE TIME TO MINS FROM THE BDGINNING
C OF THE TEST,
CALL CONVERT(NDATA(2),NDATA(3),NDATA(9),NLDOC)
CALL CONVERT(NDATA(2),NDATA(3),NDATA(10),NLDLK)
CALL CONVERT(NDATA(2),NDATA(3),NDATA(11),NLIFT)
CALL CONVERT(NDATA(2),NDATA(3),NDATA(12),NCPSN)
CALL CONVERT(NDATA(2),NDATA(3),NDATA(13),NLAND)
CALL CONVERT(NDATA(2),NDATA(3),NDATA(14),NLDUN)
C
C
C COMPURE COMPONT PARTS OF THE CRANE CYCLE
NTIME(2)=NLDLK-NLDOC
NTIME(3)=NLIFT-NLDLK
NTIME(4)=NCPSN-NLIFT
NTIME(5)=NLAND-NCPSN
NTIME(6)=NLDUN-NLAND
C
C
C IF PREVIOUS LDUN EXISTS THEN COMPUTE PART OF
C CYCLE BASED ON PREVIOUS LDUN.
NTIME(1)=-999
IF(NPREV(3) .EQ. 4RNONE) GO TO 10
CALL CONVERT(NPREV(1),NPREV(2),NPREV(3),NLDUNP)
NTIME(1)=NLDOC-NLDUNP
10 CONTINUE
C
C
RETURN
END

```

```

C      SUBROUTINE SAVE(NFACID,NDATE,NSHIFT,NLDUN,NPREV)
C
C      PURPOSE: TO RETURN PREVIOUS LDUN.  IF FIRST EVENT AT A FACILITY,
C              THERE IS NO PREVIOUS LDUN.
C
C      LOGICAL FIRST
C
C      DIMENSION NPREV(3)
C
C      DATA FIRST /6H.TRUE./
C
C      IF(.NOT.FIRST) GO TO 10
C          IFACID=4RNONE
C          FIRST=.FALSE.
10  CONTINUE
C
C
C      NPREV(3)=4RNONE
C      CHECK FOR CHANGE IN FACILITY ID
C      IF(NFACID .NE. IFACID) GO TO 20
C          NPREV(1)=IDATE
C          NPREV(2)=ISHIFT
C          NPREV(3)=ILDUN
20  CONTINUE
C
C
C      SAVE PRESENT LDUN AND FACID
C      IDATE=NDATE
C      ISHIFT=NSHIFT
C      ILDUN=NLDUN
C      IFACID=NFACID
C
C
C      RETURN
C      END

```

```

C      SUBROUTINE CONVERT(NDAY,NSHFT,NHM,MINS)
C
C      PURPOSE: CONVERTS DAYS,SHIFT,HRS AND MINS TO MINS FROM BEGINNING
C                OF THE TEST
C
C      DECODE(10,10,NDAY) ND
10  FORMAT(6X,I4)
C
C      DECODE(10,10,NSHFT) NS
C
C      DECODE(10,20,NHM) NH,NM
20  FORMAT(6X,I2,I2)
C
C
C      ADD 24 TO HRS LT 17 ON SHIFT 2
      IF(NS.EQ.2.AND.NH.LT.17) NH=NH+24
C
C
C      CALCULATE DAY OF TEST. DAY1 EQ 0.
      ND=ND-218
C
C      CALCULATE MINS.
      MINS=ND*24*60+NH*60+NM
C
C
      RETURN
      END

```

```

PROGRAM STEP15(UNIT8A,UNIT14,OUTI15,OUTC15,UNIT15,OUTPUT,
+           TAPE1=UNIT8A,TAPE2=UNIT14,TAPE3=OUTI15,
+           TAPE4=OUTC15,TAPE5=UNIT15,TAPE6=OUTPUT)

C
C
C   PURPOSE: COMBINES THE LIGHTER TIMES FILE AND THE CARGO
C           TIMES FILE INTO ONE FILE.
C
C   LOGICAL NPRINT,NSTOP,NSTOP1,MATCH,FLAG
C
C   DIMENSION KODE(2),LIGHTER(15),NCARGO(23),NEND(2)
+           ,NTEST1(4),NTEST2(4)
C
C   COMMON/TBLUNIT/NUNIT1,NUNIT2,NUNIT3,NUNIT4,NUNIT5,NUNIT6
C
C   INITIALIZE
C   NUNIT1=1
C   NUNIT2=2
C   NUNIT3=3
C   NUNIT4=4
C   NUNIT5=5
C   NUNIT6=6
C
C   NPRINT=.FALSE.
C   NSTOP= .FALSE.
C   NSTOP1=.FALSE.
C   MATCH= .FALSE.
C
C   SET KODE TO READ BOTH FILES
C   KODE(1)=1
C   KODE(2)=2
C
10  CONTINUE
C   READ LIGHTER AND/OR CARGO RECORDS.
C   CALL READ(KODE,LIGHTER,NCARGO,NEND)
C
C   CHECK FOR EOF IN ANY FILE.
C   CALL STOP(NEND,NSTOP,NSTOP1,KODE)
C   IF(NSTOP) GOTO 30
C
C   SETUP MATCH KEYS
C   CALL SETUP(LIGHTER,NCARGO,NTEST1,NTEST2)
C
C   COMPARE LIGHTER AND CARGO RECORDS.
C   CALL COMPARE(NTEST1,NTEST2,MATCH,NFAIL1,NFAIL2)
C
C   CHECK FOR A MATCH
C   IF(MATCH) GO TO 20
C
C   THE MATCH FAILED.  SELECT WHICH FILE TO READ.
C   CALL SELECT(NFAIL1,NFAIL2,KODE)

```



```

C
C      WRITE AN INCOMPLETE RECORD, IF ANY
C      CALL RPTINC(KODE,LIGHTER,NCARGO,NPRINT)
C
C      START OVER AND READ A NEW RECORD
C      GO TO 10
C
C
20 CONTINUE
  CALL SEQ(NCARGO(7),NCARGO(23),FLAG)
  IF(FLAG) GO TO 25
  KODE(1)=1
  KODE(2)=0
  NZ9=4R0000
  CALL SEQ(NZ9,0,FLAG)
  NPRINT=.FALSE.
  GO TO 10
25 CONTINUE
  A MATCH WAS FOUND
  COMPUTE TIME FROM THE BEGINNING OF THE TEST
  CALL CONVERT(NCARGO(2),NCARGO(3),NCARGO(19),NTIME)
  WRITE A COMPLETE RECORD.
  CALL RPTCOM(LIGHTER,NCARGO,NTIME,NPRINT)
C
C      SET KODE TO READ A NEW CARGO RECORD
C      KODE(1)=0
C      KODE(2)=2
C
C      READ A NEW CARGO RECORD AT THE BEGINNING
C      OF THE PROGRAM
C      GO TO 10
C
C
30 CONTINUE
  CHECK FOR EOF IN BOTH FILES.
  IF(NSTOP1) GO TO 40
C
C
C      WRITE INCOMPLETE RECORDS
C      CALL RPTINC(KODE,LIGHTER,NCARGO,NPRINT)
C      READ NEW RECORD
C      CALL READ(KODE,LIGHTER,NCARGO,NEND)
C
C      CHECK FOR EOF
C      CALL STOP(NEND,NSTOP,NSTOP1,KODE)
C
C      CONTINUE STOP PROCESS.
C      GO TO 30
C
C

```

40 CONTINUE

C
C

END


```

C      SUBROUTINE COMPARE(NTEST1,NTEST2,MATCH,NFAIL1,NFAIL2)
C
C      PURPOSE: DETERMINES IF TWO RECORDS MATCH ON FOUR MATCH KEYS
C                (I.E. FACILITY ID, LIGHTER TYPE, LIGHTER ID, AND
C                CARGO ID). IF NO MATCH, RETURNS TO THE MAIN PROGRAM
C                THE FIRST KEY WHERE THE MATCH FAILED.
C
C      LOGICAL MATCH
C
C      DIMENSION NTEST1(4),NTEST2(4)
C
C      COMPARE THE FOUR MATCH KEYS.
C      DO 10 J=1,4
C         IF(NTEST1(J) .EQ. NTEST2(J)) GO TO 10
C
C         THE MATCH FAILS.
C         MATCH=.FALSE.
C         NFAIL1=NTEST1(J)
C         NFAIL2=NTEST2(J)
C
C      RETURN
C
C      10 CONTINUE
C
C      A MATCH WAS FOUND
C      MATCH=.TRUE.
C
C      RETURN
C      END

```

```

C      SUBROUTINE SETUP(LIGHTER,NCARGO,NTEST1,NTEST2)
C
C      PURPOSE: TO CONSTRUCT TWO ARRAYS WITH FACILITY ID, LIGHTER TYPE,
C                LIGHTER ID, AND CARGO ID.
C
C      DIMENSION LIGHTER(15),NCARGO(23),NTEST1(4),NTEST2(4)
C
C      CONSTRUCT ARRAY FROM LIGHTER TIMES FILE.
C      NTEST1(1)=LIGHTER(1)
C      NTEST1(2)=LIGHTER(4)
C      NTEST1(3)=LIGHTER(5)
C      NTEST1(4)=LIGHTER(15)
C
C      CONSTRUCT ARRAY FROM CARGO TIMES FILE.
C      NTEST2(1)=NCARGO(1)
C      NTEST2(2)=NCARGO(5)
C      NTEST2(3)=NCARGO(6)
C      NTEST2(4)=NCARGO(7)
C
C      RETURN
C      END

```

```

C      SUBROUTINE SEQ(NCARGO,NSEQ,FLAG)
C
C      PURPOSE: TO MAKE SURE THE SEQ NUMBER INCREASES FOR THE SAME
C               CARGO ID.
C
C      LOGICAL FLAG
C
C      DATA NCARGOP/4R0000/
C      DATA NSEQP/0/
C
C      FLAG=.TRUE.
C
C      IF(NCARGO .EQ. NCARGOP .AND. NSEQ .LE. NSEQP) FLAG=.FALSE.
C
C      NCARGOP=NCARGO
C      NSEQP=NSEQ
C
C      RETURN
C      END

```

```

C      SUBROUTINE RPTINC(KODE,LIGHTER,NCARGO,NPRINT)
C
C      PURPOSE: WRITES AN INCOMPLETE LIGHTER OR CARGO RECORD.
C
C      LOGICAL FIRST,NPRINT
C
C      DIMENSION KODE(2),LIGHTER(15),NCARGO(23)
C
C      COMMON/TBLUNIT/NUNIT1,NUNIT2,NUNIT3,NUNIT4,NUNIT5,NUNIT6
C
C      DATA FIRST/6H.TRUE./
C
C      PRINT HEADER ON FIRST CALL
C      IF(.NOT.FIRST) GO TO 20
C          XDATE=DATE(JDUMMY)
C          XTIME=TIME(JDUMMY)
C          WRITE(NUNIT3,10) XDATE,XTIME
10      FORMAT(1H1,18HINCOMPLETE RECORDS,2(5X,A10))
C          FIRST=.FALSE.
C
C      20 CONTINUE
C
C      DETERMINE WHICH RECORD TO OUTPUT.
C      IF(KODE(1) .NE. 1) GO TO 50
C          IF(.NOT.NPRINT) GO TO 30
C          NPRINT=.FALSE.
C          RETURN
C
C      30 CONTINUE
C          NPRINT=.FALSE.
C          WRITE(NUNIT3,40) LIGHTER
40      FORMAT(1X,7HLIGHTER,10(3X,R4),3(3X,I4),2(3X,R4))
C
C          RETURN
C
C
C
C      50 CONTINUE
C          WRITE(NUNIT3,60) NCARGO
60      FORMAT(1X,6HCARGO ,9(R4,1X),6(I4,1X,R4,1X),R4,1X,I4)
C
C          RETURN
C          END

```

```

C      SUBROUTINE RPTCOM(LIGHTER,NCARGO,NTIME,NPRINT)
C
C      PURPOSE: TO WRITE A COMPLETE LIGHTER AND CARGO RECORD.
C
C      LOGICAL FIRST,NPRINT
C
C      DIMENSION LIGHTER(15),NCARGO(23)
C
C      COMMON/TBLUNIT/ NUNIT1,NUNIT2,NUNIT3,NUNIT4,NUNIT5,NUNIT6
C
C      DATA FIRST/6H.TRUE./
C
C      WRITE HEADER ON FIRST CALL
C      IF(.NOT.FIRST) GO TO 20
C          XDATE=DATE(JDUMMY)
C          XTIME=TIME(JDUMMY)
C          WRITE(NUNIT4,10) XDATE,XTIME
10      FORMAT(1H1,16HCOMPLETE RECORDS,2(5X,A10))
C          FIRST=.FALSE.
C
C
C      20 CONTINUE
C          WRITE A COMPLETE RECORD
C          WRITE(NUNIT4,30) LIGHTER,NCARGO(4),(NCARGO(J),J=9,23)
30      FORMAT(1X,R4,R3,R1,R4,R4,R1,1X,R1,1X,R1,1X,R4,1X,
C          +      R4,3(1X,I4),1X,R4,1X,R4,1X,R2,6(1X,R4,1X,I4),1X,R4,1X,R4,
C          +      1X,I1)
C
C          WRITE(NUNIT5) LIGHTER,NCARGO(4),(NCARGO(J),J=9,23),NTIME
C
C          SET KODE TO TRUE.  THE PRESENT LIGHTER RECORD WAS COMPLETE.
C          NPRINT=.TRUE.
C
C
C      RETURN
C      END

```



```

C      PROGRAM WRTRPT(TAPE1,OUTPUT,TAPE6=OUTPUT)
C
C      PURPOSE: TO READ THE SORTED LIGHTER/CARGO FILE, COMPUTE
C                STATISTICS AND WRITE A REPORT.  FOR EACH FACILITY,
C                DATE AND SHIFT, STATISTICS ARE ACCUMULATED ON
C                LIGHTER TYPES.
C
C      LOGICAL FIRST,KPGBR,SAME
C
C      DIMENSION N(32),SUMTY(9,2),SUMSQTY( 9,2),NOTYF(9),NOTYR(9)
C
C      DATA BLANK/1H /
C
C      REWIND 1
C
C      FIRST=.TRUE.
C
C      INITIALIZE BREAK KEYS
C      OFAC=BLANK
C      ODATE=BLANK
C      OSHIFT=BLANK
C
C      20 CONTINUE
C      READ A LIGHTER/CARGO
C      READ(1) N
C      IF( EOF(1) ) 30,40
C      30 CONTINUE
C      CALL CHECK(N,SAME)
C      CALL PRNTST(SUMTY,SUMSQTY,NOTYF,NOTYR)
C      STOP
C
C      40 CONTINUE
C      CALL CHECK(N,SAME)
C      KPGBR=.FALSE.
C
C      CHECK FOR PAGE BREAK ON FACILITY,DATE,OR SHIFT.
C      IF( (N(1) .NE. OFAC) .OR. (N(2) .NE. ODATE) .OR.
+      (N(3) .NE. OSHIFT) ) CALL PGBRK(FIRST,OFAC,ODATE,OSHIFT,N,
+      KPGBR,OLTRTY,OFR,SUMTY,
+      SUMSQTY,NOTYF,NOTYR,SAME)
C      IF(KPGBR) GO TO 20
C
C      CHECK FOR LIGHTER TYPE OR FORWARD OR RETROGRADE BREAK
C      IF( N(4) .EQ. OLTRTY) GO TO 50
C      OLTRTY=N(4)
C      OFR=N(8)
C
C      CALL PRNTST(SUMTY,SUMSQTY,NOTYF,NOTYR)

```

```

      CALL INTY(SUMTY,SUMSQTY,NOTYF,NOTYR)
      CALL ACCTY(SUMTY,SUMSQTY,NOTYF,NOTYR,N,SAME)
      CALL PRNIND(N,SAME)
      GO TO 20
50  CONTINUE
      IF( N(8) .EQ. OFR) GOTO 60
      OFR=N(8)
      CALL ACCTY(SUMTY,SUMSQTY,NOTYF,NOTYR,N,SAME)
      CALL PRNIND(N,SAME)
      GO TO 20

```

C
C

```

60  CONTINUE
      CALL PRNIND(N,SAME)
      CALL ACCTY(SUMTY,SUMSQTY,NOTYF,NOTYR,N,SAME)
      GO TO 20

```

C
C

END

```

SUBROUTINE ACCTY(SUMTY,SUMSQTY,NOTYF,NOTYR,N,SAME)
C
C PURPOSE: COMPUTE AND ACCUMULATE LIGHTER AND CARGO TYPE
C + STATISTICS.
C
C LOGICAL SAME
C
C DIMENSION SUMTY(9,2),SUMSQTY(9,2),N(32),L(9),NOTYR(9),NOTYF(9)
C
C PUT NUMBERS FOR STATISTICS INTO ONE ARRAY "L"
L(1)=N(11)
L(2)=N(12)
L(3)=N(13)
L(4)=N(18)
L(5)=N(20)
L(6)=N(22)
L(7)=N(24)
L(8)=N(26)
L(9)=N(28)
C
C NSTART=1
C IF(SAME) NSTART=4
C
C CHECK FOR FORWARD OR RETROGRADE.
C IF( N(8) .EQ. 4R F ) GO TO 30
C
C RETROGRADE STATISTICS
C DO 20 J=NSTART,9
C IF( L(J) .EQ. -999 .OR. N(31) .NE. 1 ) GO TO 10
C NOTYR(J)=NOTYR(J)+1
C SUMTY(J,2)=SUMTY(J,2) + L(J)
C SUMSQTY(J,2)=SUMSQTY(J,2) + L(J) * L(J)
10 CONTINUE
20 CONTINUE
C
C RETURN
C
C
C
C 30 CONTINUE
C FORWARD STATISTICS
C DO 50 J=NSTART,9
C IF( L(J) .EQ. -999 .OR. N(31) .NE. 1 ) GO TO 40
C NOTYF(J)=NOTYF(J)+1
C SUMTY(J,1)=SUMTY(J,1) + L(J)
C SUMSQTY(J,1)=SUMSQTY(J,1) + L(J) * L(J)
40 CONTINUE
50 CONTINUE
C
C
C RETURN
C END

```

```

SUBROUTINE INTY(SUMTY,SUMSQTY,NOTYF,NOTYR)
C
C PURPOSE: INITIALIZE STATISTICS PARAMETERS.
C DIMENSION NOTYF(9),NOTYR(9),SUMTY(9,2),SUMSQTY(9,2)
C
C DO 10 J=1,9
C   NOTYF(J)=0
C   NOTYR(J)=0
10 CONTINUE
C
C DO 30 I=1,9
C   DO 20 J=1,2
C     SUMTY(I,J)=0
C     SUMSQTY(I,J)=0
20 CONTINUE
30 CONTINUE
C
C RETURN
END

```



```
C   ACCUMULATE LIGHTER TYPE STATISTICS
C   CALL ACCTY(SUMTY,SUMSQTY,NOTYF,NOTYR,N,SAME)
C
C   OFAC=N(1)
C   ODATE=N(2)
C   OSHIFT=N(3)
C   KPGBR=.TRUE.
C
C   RETURN
C   END
```

```

SUBROUTINE PRNIND(N,SAME)
C
C PURPOSE: TO WRITE INDIVIDUAL LIGHTER/CARGO RECORDS
C
C LOGICAL SAME
C
C DIMENSION N(32)
C
C IF DELAY RECORD
C IF( N(31) .EQ. 1 ) GO TO 20
C THEN PRINT DELAY PART OF THE RECORD
C WRITE(6,10 ) N(29),N(30)
10 FORMAT(116X,R4,1X,R4)
C
C RETURN
C
C ELSE PRINT THE COMPLETE RECORD
20 CONTINUE
C DECODE(10,30,N(6)) N6
30 FORMAT(6X,R1,3X)
C DECODE(10,40,N(7)) N7
40 FORMAT(6X,R2,2X)
C DECODE(10,40,N(16)) N16
C
C IF(SAME) GO TO 60
C
C WRITE(6,50) (N(I),I=4,5),N6,N7,(N(I),I=8,15),N16,(N(I),I=17,30)
50 FORMAT(1X,R4,1X,R4,1X,R1,1X,R2,1X,R1,1X,R4,1X,R4,1X,I4,1X,I4,
+ 1X,I4,1X,R4,1X,R4,1X,R2,1X,R4,1X,I4,1X,R4,1X,I4,1X,R4,1X,
+ I4,1X,R4,1X,I4,1X,R4,1X,I4,1X,R4,1X,I4,1X,R4,1X,R4)
C
C RETURN
C
60 CONTINUE
C WRITE(6,70) (N(I),I=14,15),N16,(N(I),I=17,30)
70 FORMAT(43X,R4,1X,R4,1X,R2,1X,R4,1X,I4,1X,R4,1X,
+ I4,1X,R4,1X,I4,1X,R4,1X,I4,1X,R4,1X,
+ I4,1X,R4,1X,I4,1X,R4,1X,R4)
C
C RETURN
C
END

```



```

SUBROUTINE PRNTST(SUMTY,SUMSQTY,NOTYF,NOTYR)
C
C
C   PRUPOSE; COMPUTE AND PRINT LIGHTER TYPE STATISTICS.
C
C   DIMENSION SUMTY(9,2),SUMSQTY(9,2),XBAR(9,2),SD(9,2),
+     NOTYF(9),NOTYR(9)
C
C   DO 30 J=1,2
C     DO 20 I=1,9
C       N=NOTYF(I)
C       IF(J.EQ. 2) N=NOTYR(I)
C       IF(N.NE. 0) GO TO 10
C       XBAR(I,J)=0.
C       SD(I,J)=0.
C       GO TO 20
10    CONTINUE
C       XBAR(I,J)=SUMTY(I,J)/N
C       SD(I,J)=SUMSQTY(I,J)/N - XBAR(I,J)*XBAR(I,J)
20    CONTINUE
30    CONTINUE
C
C   DO 50 I=1,9
C     DO 40 J=1,2
C       Z9=SD(I,J)
C       SD(I,J)=SQRT(Z9)
40    CONTINUE
50    CONTINUE
C
C
C   WRITE(6,60)
60  FORMAT(3X,7HFORWARD)
C   WRITE(6,70) NOTYF
70  FORMAT(10X,1HN,17X,I4,1X,I4,1X,I4,19X,6(I4,6X))
C   WRITE(6,80) ( XBAR(I,1),I=1,9),(SD(J,1),J=1,9)
80  FORMAT(10X,4HMEAN,13X,3(F5.1),18X,6(F5.1,5X)/
+     10X,4HSD ,13X,3(F5.1),18X,6(F5.1,5X))
C
C
C   WRITE(6,90)
90  FORMAT(3X,10HRETROGRADE)
C   WRITE(6,70) NOTYR
C   WRITE(6,80) ( XBAR(I,2),I=1,9),(SD(J,2),J=1,9)
C
C
C   RETURN
C   END

```

```

C      SUBROUTINE CHECK(N,SAME)
C
C      PURPOSE: TO DETERMINE IF PRESENT LIGHTER RECORD IS THE SAME
C                AS THE PREVIOUS LIGHTER RECORD.
C
C      LOGICAL SAME
C
C      DIMENSION N(32),NOLD(10)
C
C      DATA NOLD/10*1/
C
C      SAME=.TRUE.
C
C      DO 10 J=1,10
C         IF( N(J) .EQ. NOLD(J) ) GO TO 10
C         DIFFERENT LIGHTER RECORD FOUND
C         SAME=.FALSE.
10    CONTINUE
C
C      SAVE PRESENT LIGHTER PARAMETERS
C      DO 20 J=1,10
C         NOLD(J)=N(J)
20    CONTINUE
C
C      RETURN
C      END

```

APPENDIX D

CRANE CYCLE TIMES
PROGRAM LISTINGS

```

LSTGT,CM60000,SPUU,T500.GST
TASK(TN=LOTS,TA=12345,OS=ORISPM,TR=TS,PI=GST)
COMMENT.***** CTL. STMTS. ON GENJ4CYCLEDATA *****
ATTACH(S2K,S2K260,ID=SYS2000,MR=1)
REQUEST(RETRS2K,*PF)
S2K(CR=77)
ATTACH(MAGIC,MAGICNUMBERGENERATOROBJECT,ID=LOTS,MR=1)
REQUEST(RETRMGC,*PF)
MAGIC(RETRS2K,RETRMGC)
REWIND(RETRMGC)
REQUEST(RETRSTD,*PF)
FILE(RETRMGC,MRL=110,RT=Z,BT=C,FL=110)
FILE(RETRSTD,MRL=110,RT=Z,BT=C,FL=110)
SORTMRG.      SORT RETRMGC TO RETRSTD BY MAGIC NO.
CATALOG(RETRSTD,JLMT4ACARDF,ID=LOTS)
EXIT.
CATALOG(RETRS2K,JLMT4ACARDFUNSORTED,ID=LOTS)
*EOR
USER,LOTS; SHARED DBN IS JLMT4A;
REPORT FILE IS RETRS2K;
LI/TITLE L(4),L(3),L(1),L(4),L(1),L(4),L(4),L(4),L(4),L(4),L(2),
L(4),L(4)/
C1010,C1110,C1120,C1305,C1310,C1320,C1325,
C1330,C1335,C1340,C1365,C1410,C1420,
OB C1010,C1110,C1120,C1420
WH ((C1410 EQ LDOC OR C1410 EQ DELY)
AND (C1010 EQ COD OR C1010 EQ TCDF OR C1010 EQ ADP
OR C1010 EQ ECWY OR C1010 EQ BBCS OR C1010 EQ EDP))
AND (C1370 EQ C);
*EOR
SORT
FILE,SORT=RETRMGC,OUTPUT=RETRSTD
FIELD,FACID(4,4,DISPLAY),DATE(11,3,DISPLAY),SHIFT(17,1,DISPLAY),
,CARID(21,4,DISPLAY),CY(28,1,DISPLAY),DISCAR(32,4,DISPLAY),
,DISID(39,4,DISPLAY),RECCAR(46,4,DISPLAY),RECID(53,4,DISPLAY),
,DEVICE(60,4,DISPLAY),UNK(67,2,DISPLAY),NAME(72,4,DISPLAY),
,TIME(79,4,DISPLAY),NTIME(86,6,DISPLAY)
KEY,FACID(A,DISPLAY),NTIME(A,DISPLAY),NAME(D,DISPLAY)
END
*EOR
*EOF

```

```

PROGRAM MAGIC(TAPE1, TAPE2, OUTPUT, TAPE6=OUTPUT)

DIMENSION N(14)

REWIND 1

READ(1,10)
10 FORMAT(/)

12 CONTINUE
  READ(1,20) (N(I),I=1,13)
  IF(EOF(1)) 50,15
15 CONTINUE
20 FORMAT(3X,R4,3X,R3,3X,R1,3X,R4,3X,R1,5(3X,R4),3X,R2,
+       2(3X,R4))
  CALL CONVERT(N(2),N(3),N(13),N(14),NTIME)
  WRITE(2,30) (N(I),I=1,12), NTIME,N(14)
30 FORMAT(3X,R4,3X,R3,3X,R1,3X,R4,3X,R1,5(3X,R4),3X,R2,3X,
+       R4,3X,I4,3X,I6)
  GO TO 12
50 CONTINUE

END

```

```

SUBROUTINE CONVERT(NDAY,NSHFT,NHM,MINS,NTIME)

*   PURPOSE: CONVERTS DAYS,SHIFT,HRS AND MINS TO MINS FROM BEGINNING
*           OF THE TEST

      DECODE(10,10,NDAY) ND
10  FORMAT(7X,I3)

      DECODE(10,15,NSHFT) NS
15  FORMAT(9X,I1)

      DECODE(10,20,NHM) NH,NM
20  FORMAT(6X,I2,I2)

*   ADD 24 TO HRS LT 17 ON SHIFT 2
      IF(NS.EQ.2.AND.NH.LT.17) NH=NH+24

*   CALCULATE DAY OF TEST. DAY1 EQ 1.
      ND=ND-217

*   CALCULATE MINS.  ADD 10000 SO ALL MINS WILL BE 5 CHARCTERS
*   LONG.  THIS IS IMPORTANT FOR THE BCD SORT WHICH SORTS ON BLANKS.
*   IN THE BCD SORT A BLANK IS GT A NINE.
      MINS=ND*24*60+NH*60+NM+10000

*   ADD 24 TO HRS. < 7 ON SHIFT 2
      NTIME = NH * 100 + NM
      IF (NS .EQ. 2 .A. NH .LT. 7) NTIME = (NH+24) * 100 + NM

      RETURN
      END

```

1 2 3 4 5 6 7 8 9

```
overlay('cyclovr',0,0)
program cyclops (cardf, scfile, dbintr, scintr, report, output,
  tape1=cardf, tape2=scfile, tape3=dbintr, tape4=scintr,
  tape5=report, tape6=output, csidts=200, tape7=csidts)
```

```
#-----
#      Call overlay(1,0) and overlay(2,0) to set up "data base
#      intermediate file" (DBINTR), and "selection criteria intermediate
#      file" (SCINTR), respectively.
#
```

```
#      Call overlay(3,0) to produce a statistical report on selected
#      cycles, and/or the selected cycles themselves, according to the
#      setting of sense switches 1, 2, and 3--as indicated below.
#-----
```

```
#####
# This program was developed by Gary S. Trujillo #
#####
```

```
#-----
implicit integer (a-z)
common /runinfo/ cdate, ctime
#-----
```

```
call sswtch(1, sw1)      # produce statistics report
call sswtch(2, sw2)      # print cargo cycle data
call sswtch(3, sw3)      # print exp1, SCs & sel. cycle data
```

```
if (sw1==2 & sw2==2 & sw3==2) {
  call remark('error--no options selected via sense switch')
  stop
}
```

```
call time(ctime)          # set time & date from system
call date(cdate)
```

```
rewind NREPORT ; write (NREPORT,10) ctime, cdate
                  write (NERRRPT,10) ctime, cdate
```

```
call overlay('cyclovr',1,0) # create DBINTR file
call overlay('cyclovr',2,0) # create SCINTR file
call overlay('cyclovr',3,0) # generate statistical reports
```

```
10 format('1beginning execution ',2a10)
```

```
end
```

1 2 3 4 5 6 7 8 9

overlay('cyclovr',1,0)

program crdbint

create "data base intermediate file"

#-----
implicit integer (a-z)

common /dbinfo/ dbname, dbdate, dbtime

common /runinfo/ cdate, ctime

integer padding[6]

data padding/6*0/

data filename //'dbintr'/

data boscode /BOG/
#-----

if (setidwc(dummy) == ERR) { # input cargo ID's & weight codes

call remark('error--cargo id weight file is null')

stop

}

rewind NDBINTR

set one record from cargo activity

raw data file

rstatus = rdcardf(fcltyid, oprdate, oprshft, dirmvmt,
cstype, sselfdev, operid, wshtcat, cystart, cydurat,
spclxpt, lstrseq, cyinter)

if (rstatus == EOF)

return

if cargo activities file is null,

return without touching DBINTR

if (rstatus ~= BOG) {

write (NERRRPT,10)

first record not beginning-of-group

call remark('error in crdbint in reading cargo data file')

stop

}

write (NDBINTR) filename,

name of file

dbname, cdate, ctime,

creation date and time

dbdate, dbtime

data base file creation info.

while (rstatus ~= EOF) {

if (rstatus == BOG)

write header for each new group

write (NDBINTR) boscode, fcltyid, oprdate, oprshft, padding

write (NDBINTR) dirmvmt, cstype, operid, sselfdev,

wshtcat, cystart, cydurat, spclxpt, lstrseq, cyinter

rstatus = rdcardf(fcltyid, oprdate, oprshft, dirmvmt,
cstype, sselfdev, operid, wshtcat, cystart, cydurat,
spclxpt, lstrseq, cyinter)

}

10/16/78

15.06.05.

CRDBINT

PAGE NO. 2

1 2 3 4 5 6 7 8 9

rewind NDBINTR

#-----
10 format('0first record returned by rdcardf not beginning-of-group')

end

1 2 3 4 5 6 7 8 9

Read "Cargo Activities Raw Data File" (CARDF)

```
#-----
integer function rdcарdf (fp1,fp2,fp3,fp4,fp5,fp6,fp7,fp8,fp9,
                        fp10,fp11,fp12,fp13)
```

implicit integer (a-z)

logical fstcall, endfile, clasify, newsync

common /dbinfo/ dbname, dbdate, dbtime

common /ncdrec/ ncdrec # no. of CARDF record

data wstcode, carsid, dxident, rxident, sldev,

operid, evtname, evttime, lstrbrk /9 * 0/,

fstcall/.true./, newsync/.false./

```
#-----
if (fstcall) { # read file header on first call
```

rewind NCARDF

read (NCARDF,15) dbname, dbdate, dbtime

if (endfile(NCARDF)) {

write (NERRRPT,20)

cargo activity raw data file null--

rdcardf = EOF

return

bypass CARDF creation

}

if (dbname == ' ' ! dbdate == ' ' ! dbtime == ' ') {

write (NERRRPT,12) dbname, dbdate, dbtime

call remark('improper header on cargo activity raw data file')

stop

}

fstcall = .false.

write (NREPORT,25) dbdate, dbtime

write (NERRRPT,25) dbdate, dbtime

}

rdcardf = OK

set default return code

fp13 = UNINTER

cycle unint. unless delay found

while (.true.) {

if (evtname ~= 'dely') {

unless last record was for

prvwtd = wstcode

a delay event,

prvcid = carsid

save all important values

prvdxid = dxident

prvrxid = rxident

prvsldv = sldev

prvopid = operltr

prvevnm = evtname

prvevtm = evttime

prvltbk = lstrbrk

}

read (NCARDF,10) facid, opdate, oprshft, carsid, cscycle,

1 2 3 4 5 6 7 8 9

dxtype, dxident, rxttype, rxident,
sldev, operid, evtname, evttime, ncdrec

```
if (endfile(NCARD)) {
  rdcарdf = EOF          # signal end-of-file
  return                # to caller
}
```

```
# CLASIFY determines cargo direction
# of-movement(fp4), cycle type(fp5),
# and lighter break code based on
# type of discharging & receiving
# transporters and facility ID
```

```
if (~clasify(facid, dxtype, rxttype, # input parameters
            fp4, fp5, lstrbrk)) {    # output parameters
  write (NERRRPT,50) ncdrec, facid, dxtype, rxttype
  newsync = .true.                  # unable to classify record
  next
}
```

```
wstcode = wshtcat(carsid)
operltr = cvoprid(operid) # convert numeric code to alpha
if (operltr == ERR)
  write (NERRRPT,70) operid, ncdrec
```

```
if (wstcode == ERR) {
  write (NERRRPT,30) ncdrec, carsid, facid, oprdate, oprshft
  newsync = .true.      # skip cycle if cargo ID
  next                  # is invalid
}
```

```
evttime = minutes(evttime) # convert event time to minutes
```

```
if (newsync) {          # do not calculate cycle
                        # if error on previous record
  newsync = .false.
  if (facid == prvfcid # but if current record
      & oprdate == prvdate # begins a new group
      & oprshft == prvshft) # process it as usual
    next
}
```

```
if (facid ~= prvfcid    # check for new group
    | oprdate ~= prvdate
    | oprshft ~= prvshft) { # *** NEW GROUP ***
```

```
  if (evtname == 'dely') # if 1st record of new group
    next                 # contains a delay event,
                        # just ignore it
```

```
  if (rdcardf == BOG)    # if previous record also had a
```

1 2 3 4 5 6 7 8 9

```

# different facility ID, date
# or shift code from the record
# immediately preceding it,
# something is probably wrong.

```

```

write (NERRRPT,40) prvfcd, prvdate, prvshft, facid,
oprdate, oprshft, ncdrec

```

```

fp1 = prvfcd = facid
fp2 = prvdate = oprdate
fp3 = prvshft = oprshft
rdcardf = BOG
lstrseq = 1

```

```

# arrange that 1st cycle time
# will have lighter seq. of 1

```

```

}

```

```

#-----
else {                                     # *** NOT NEW GROUP ***

# set SC parameters based on data from previous record

if (evtname == 'dely') { # if cargo activity set
    fp13 = INTER          # contains "delay" activity code,
    next                # mark cycle as "interrupted"
}
if (carsid == prvcid) {
    write (NERRRPT,60) ncdrec, carsid, facid, oprdate, oprshft
    next                # duplicate record--ignore it
}
if (lstrbrk ~= prvlbrk) # start new lighter sequence
    lstrseq = 0          # if change of direction
fp6 = prvsldv
fp7 = prvopid
fp8 = prvwtd
fp9 = prvevtm
fp10 = evttime - prvevtm
if (fp10 < 0)
    write (NERRRPT,80) fp10, ncdrec # negative cycle time

if (dxtype == 'lshb' ; rxtype == 'lshb')
    fp11 = LSHB
else if (dxtype == 'sbbg' ; rxtype == 'sbbg')
    fp11 = SBBG
else fp11 = UNDEF

if (lstrbrk == DISCHARGE) { # if breaking on new discharging
    if (dxident ~= prvdxd) # lighter, see if it has been
        lstrseq = 0        # encountered
}
else if (lstrbrk == RECEIVE) {
    if (rxident ~= prvrxd) # same test for receiving
        lstrseq = 0        # lighter
}
}

```

1 2 3 4 5 6 7 8 9

```

else
    lstrseq = UNDEF
    fp12 = lstrseq
    lstrseq = lstrseq + 1
    return
}
}

```

```

#-----
10 format(3x, a4,      # facid
          3x, a3,      # oprdate
          3x, a1,      # oprshift
          3x, a4,      # cardid
          3x, a1,      # cscycle
          5(3x, a4),    # dxtype, dxident, rxtype, rxident, sldev
          3x, r2,      # operid
          3x, a4,      # evtname
          3x, i4,      # evttime
          3x, i6 )     # ncdrec

```

```

12 format('0improper header on cargo activity raw data file',
          // file name given as      ',a10,
          // creation date given as ',a10,
          // creation time given as ',a10)

```

```

15 format(a10,4x,2(a10,5x)) // # header record

```

```

20 format('0cargo activities raw data file is null',
          '--bypassing intermediate file creation')

```

```

25 format('0creating intermediate file from raw data file of '2a10)

```

```

30 format('0record no. ',i5,' of cargo activities raw data file ',
          'contains invalid cargo identifier (',a4,') ',
          / ' facility id= ',a10,' date= ',a9,' shift= ',a1)

```

```

40 format('0rdcardf detected two consecutive changes in facility id ',
          'date, or shift codes',2(/5x,3a10),10x,'record no. ',i5)

```

```

50 format(' record no. ',i5,' of cargo activities raw data file',
          ' contains an invalid code.',
          / ' facility id= ',a10,' discharging transporter type= ',a10,
          ' receiving transporter type= ',a10)

```

```

60 format('0record no. ',i5,' of cargo activities raw data file ',
          ' appears to duplicate the immediately preceding record',
          / ' cargo id= ',a10,' facility id= ',a10,' date= ',a9,
          ' shift= ',a1)

```

10/16/78

15.22.48.

RDCARDF

PAGE NO. 5

1 2 3 4 5 6 7 8 9

70 format('Oinvalid operator id "',r2,'" in record no. ',
i5,' of cargo activities raw data file')

80 format('Onegative cycle time ',i5,
' detected by r:cardf for record no. ',i5)

end

1 2 3 4 5 6 7 8 9

```
integer function wghtcat(csid) # look up & return wt. code for CGID
```

```
implicit integer (a-z)
```

```
integer carstbl[CRGTBLSIZ]
```

```
integer wshttbl[CRGTBLSIZ]
```

```
real weight
```

```
logical endfile
```

```
common /wstcom/ wccount[7]
```

```
data prvcid /0/
```

```
shiftlr(id) = id/2 .a. 3777 77777 77777 77777b # shift 1 bit right
# & mask off sign
```

```
#-----
# for (i=1; i<=lmttbl; i=i+1) {
```

```
  i = 1
```

```
  idtmp = shiftlr(csid) # eliminate sign bit for comparison
```

```
  while(i <= lmttbl) {
```

```
    if (idtmp == shiftlr(carstbl[i])) {
```

```
      wghtcat = wshttbl[i]
```

```
      return
```

```
    }
```

```
    if (csid < carstbl[i])
```

```
      break
```

```
    i = i + 1
```

```
  }
```

```
  wghtcat = ERR # cargo ID not found in table
```

```
  return
```

```
#-----
entry setidwc # input weights & convert to wt. codes
rewind NCGIDTS
```

```
do i = 1, CRGTBLSIZ
```

```
{
```

```
  read (NCGIDTS,10) carstbl[i], weight, longltr
```

```
  if (endfile(NCGIDTS)) {
```

```
    lmttbl = i - 1
```

```
    if (lmttbl == 0) {
```

```
      write (NERRRPT,40) # cargo id file is null
```

```
      call remark('null cargo id file detected by setidwc')
```

```
      wghtcat = ERR
```

```
    }
```

```
    write(NREPORT,60) (J, wccount[J], J=1,6)
```

```
    write(NREPORT,61) wccount[7]
```

```
    return
```

```
  }
```

```
  wshttbl[i] = wcatset(weight, longltr)
```

```
  if (carstbl[i] <= prvcid) {
```

```
    write (NERRRPT,20) carstbl[i], i
```

```
    wghtcat = ERR
```

1 2 3 4 5 6 7 8 9

```
        return
    }
    prvcid = carstbl[i]
}
write (NERRRPT,30)
call remark('cargo/weight table overflow in setidwc')
wshtcat = ERR
```

10 format(a4,1x,f3.1,1x,a1)

20 format('0cargo ident ',a4,' out of sequence--',
 'table element no. ',i4)

30 format('0cargo ident/weight-code table overflow.')

40 format('0cargo id file is null.')

60 format('/(' count for weight category ',r1,'=',i4))

61 format('0count of undefined weights= ',i4)

end

1 2 3 4 5 6 7 8 9

```
# Determine weight category of container of weight WEIGHT
# (LONGLTR == '1' indicates "long container").
```

```
#-----
integer function wcatset(weight, longltr)
```

```
implicit integer (a-z)
real weight
common /wstcom/ wccount[7]
data wccount /7*0/
```

```
#-----
```

```
wtlbs = weight * 2000           # convert from short tons to pounds
```

```
if (longltr == '1') {
  if (wtlbs < 43001)
    wcat = 5
  else if (wtlbs > 50000)
    wcat = 6
  else
    wcat = 7                    # no recognizable weight category
```

```
} else {
  if (wtlbs < 6801)
    wcat = 1
  else if (wtlbs > 10800 & wtlbs < 12801)
    wcat = 2
  else if (wtlbs > 19800 & wtlbs < 23801)
    wcat = 3
  else if (wtlbs > 43800 & wtlbs < 45801)
    wcat = 4
  else
    wcat = 7                    # no recognizable weight category
```

```
}
```

```
wcatset = shift(1,wcat-1)      # return a 1-bit mask
```

```
wccount[wcat] = wccount[wcat] + 1  # increment appropriate counter
```

```
end
```

1 2 3 4 5 6 7 8 9

convert operator ID from 2-digit number to a
letter code in the range A - L

integer function cvoprid(operatorid)

implicit integer (a-z)

integer opertbl[12]

data opertbl /2r01, 2r02, 2r03, 2r04, 2r05, 2r06,
2r07, 2r08, 2r09, 2r10, 2r11, 2r12/

#

if (operatorid == 2555b) { # (2ru)

cvoprid = UNDEF

return

}

do i = 1, 12

{

if (operatorid == opertbl[i]) {

cvoprid = i

return

}

}

cvoprid = ERR

return

end

1 2 3 4 5 6 7 8 9

```
# Determine direction-of-movement, cycle type, and dsch/recv break
# code for indicated discharger, receiver, facility ID combination
#-----
logical function clasify (facilid, dschxpt, recvxpt, # input params.
                        dirmvmt, cycotype, xptbrk) # output params.

implicit integer (a-z)
common /ncdrec/ ncdrec
#-----

clasify = .true.                                # optimistic default
dirmvmt = cycotype = xptbrk = UNDEF

# determine fccode, dxcode, and rxcode

if (fcltype(facilid, fccode) == ERR) {           # facility type
  write (NERRRPT,2) facilid, ncdrec              # invalid facility code
  go to 90
}
if (xpttype(dschxpt, dxcode) == ERR) {           # dsch. xptr. type
  write (NERRRPT,4) dschxpt, ncdrec              # invalid DX code
  go to 90
}
if (xpttype(recvxpt, rxcode) == ERR) {           # recv. xptr. type
  write (NERRRPT,6) recvxpt, ncdrec              # invalid RX code
  go to 90
}
}
#-----
if (fccode == LANL) {                             # some inland facility
  if (dxcode == LIGHTER) {                         # cargo discharged by lighter
    if (rxcode == STOR) {                          # & received into storage
      dirmvmt = OFFLOAD
      cycotype = 2
      xptbrk = DISCHARGE
    } else if (rxcode == LANDV) {                  # & received by land vehicle
      dirmvmt = OFFLOAD
      cycotype = 1
      xptbrk = DISCHARGE
    } else {
      write (NERRRPT,8) recvxpt, ncdrec            # invalid receiver
      go to 90
    }
  } else if (dxcode == STOR) {                     # cargo removed from storage
    if (rxcode == LIGHTER) {
      dirmvmt = RETROGD
      cycotype = 3
      xptbrk = RECEIVE
    } else if (rxcode == STOR) {
      cycotype = 4
    } else if (rxcode == LANDV) {

```

1 2 3 4 5 6 7 8 9

```

        dirmvmt = OFFLOAD
        cycstype = 3
    } else {
        write (NERRRPT,10) recvxpt, ncdrec # invalid receiver
        go to 90
    }
} else if (dxcode == LANDV) { # cargo discharged by land vehicle
    if (rxcode == LIGHTER) {
        dirmvmt = RETROGD
        cycstype = 1
        xptbrk = RECEIVE
    } else if (rxcode == STOR) {
        dirmvmt = OFFLOAD
        cycstype = 2
    } else {
        write (NERRRPT,12) recvxpt, ncdrec # invalid receiver
        go to 90
    }
} else {
    write (NERRRPT,14) dschxpt, ncdrec # invalid discharger
    go to 90
}
}

-----
else if (fccode == SEA) { # COD or TCDF
    if (dxcode == LIGHTER) {
        if (rxcode == STOR) {
            dirmvmt = RETROGD
            cycstype = 2
            xptbrk = DISCHARGE
        } else if (rxcode == NSSC) {
            dirmvmt = RETROGD
            cycstype = 1
            xptbrk = DISCHARGE
        } else {
            write (NERRRPT,8) recvxpt, ncdrec # invalid receiver
            go to 90
        }
    }
} else if (dxcode == STOR) {
    if (rxcode == LIGHTER) {
        dirmvmt = OFFLOAD
        cycstype = 3
        xptbrk = RECEIVE
    } else if (rxcode == STOR) {
        cycstype = 4
    } else if (rxcode == NSSC) {
        dirmvmt = RETROGD
        cycstype = 3
    } else {
        write (NERRRPT,10) recvxpt, ncdrec # invalid receiver

```

MD-A131 715

JOINT LOGISTICS-OVER-THE-SHORE (LOTS) MAIN TEST
AUTOMATED DATA BASE REDUCTION PROGRAMS(U) ORI INC
SILVER SPRING MD H CASEY ET AL. MAR 79 ORI-TR-1477

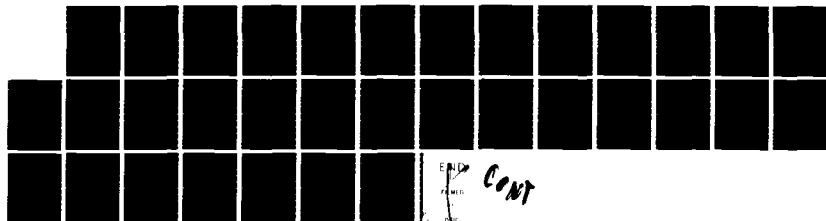
5/6

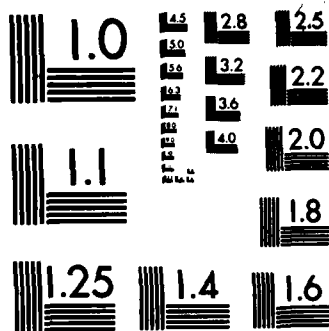
UNCLASSIFIED

MDA903-75-C-0016

F/G 9/2

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```

1  2  3  4  5  6  7  8  9
      so to 90
    }
  } else if (dxcode == NSSC) {
    if (rxcode == LIGHTER) {
      dirvmmt = OFFLOAD
      cycstype = 1
      xptbrk = RECEIVE
    } else if (rxcode == STOR) {
      dirvmmt = OFFLOAD
      cycstype = 2
    } else {
      write (NERRRPT,16) recvxpt, ncdrec # invalid receiver
      so to 90
    }
  } else {
    write (NERRRPT,14) dschxpt, ncdrec # invalid discharger
    so to 90
  }
} else {
  write (NERRRPT,18) facilid, ncdrec
90  clasify = .false.
}
#-----
2 format('0invalid facility code ',a4,' in cardf record no. ',i5)
4 format('0invalid dx code ',a4,' in cardf record no. ',i5)
6 format('0invalid rx code ',a4,' in cardf record no. ',i5)
8 format('0invalid receiver code ',a4,
' for cargo discharged from lighter in cardf record no. ',i5)
10 format('0invalid receiver code ',a4,
' for cargo discharged from storage in cardf record no. ',i5)
12 format('0invalid receiver code ',a4,
' for cargo discharged from land vehicle in cardf record no. ',
i5)
14 format('0invalid discharger code ',a4,
' for sea facility in cardf record no. ',i5)
16 format('0invalid receiver code ',a4,
' for cargo discharged from nssc in cardf record no. ',i5)
18 format('0invalid code ',a4,' returned by fclype in record no. ',i5,
' -- fatal error, execution terminates')

end

```

1 2 3 4 5 6 7 8 9

integer function fcltype (fcname, ft) # determine facility type

implicit integer (a-z)

```
-----
if (fcname == 'cod' ! fcname == 'tcd') {
    fcltype = ft = SEA
    return
}

if (fcname=='adr' ! fcname=='ecwy' ! fcname=='bbcs' ! fcname=='edr'){
    fcltype = ft = LAND
    return
}

fcltype = ft = ERR

end
```


1 2 3 4 5 6 7 8 9

integer function xpttype (xptname, xt)

determine type of transporter
(STOR, NSSC, LANDV, or LIGHTER)

implicit integer (a-z)

if (xptname == 'stor') {
 xpttype = xt = STOR
 return
}if (xptname == 'nssc') {
 xpttype = xt = NSSC
 return
}if (xptname == '5mc' | xptname == '5yt' | xptname == '5s&p') {
 xpttype = xt = LANDV
 return
}if (xptname == 'lcm8' | xptname == 'lcv' | xptname == 'lcv'
 | xptname == 'lc15' | xptname == 'sbbs' | xptname == 'lc60'
 | xptname == 'cfrv' | xptname == 'lshb') {

 xpttype = xt = LIGHTER
 return
}xpttype = xt = ERR
return

end

1 2 3 4 5 6 7 8 9

```
overlay('cyclovr',2,0)
```

```
Program crscint          # create selection criteria intermediate file
```

```
implicit integer (a-z)
```

```
logical endfile, wstmskr, bstblok
```

```
integer wc[6], title[4]
```

```
integer selcrit[12]
```

```
integer sname[2,12]
```

```
integer bysentb, atlimit[6], attrtbl[13,6]
```

```
real bktsize
```

```
integer auxinfo[8]          # MUST BE 8 WORDS
```

```
equivalence (auxinfo[1],nbucket), # add'l. info. to be written
```

```
            (auxinfo[2],bktsize), # to SCINTR file
```

```
            (auxinfo[3],prthist),
```

```
            (auxinfo[4],nscrec),
```

```
            (auxinfo[5],title)    # N.B. TITLE is four (4) words
```

```
# -----
```

```
# Each element of the ATLIMIT array indicates the number  
# of elements in the corresponding row of the ATTRTBL array.
```

```
                                # attribute-table limits (for "BY"-codes)
```

data atlimit	/	2,	# direction-of-movement
		2,	# cycle interruption
		4,	# sling/lifting device
		13,	# operator ID
		4,	# cycle type
		7 /	# weight category

```
# -----
```

```
# The ATTRTBL array defines the legitimate values for  
# those selection criteria categories which can be selected to  
# automatically range through their values (known as the "BY"  
# feature in System 2000).
```

data (attrtbl[i,1],i=1,2)	/OFFLOAD, RETROGD/,
(attrtbl[i,2],i=1,2)	/INTER, UNINTER/,
(attrtbl[i,3],i=1,4)	//hosa', 'manl', 'sling', 'u'//,
(attrtbl[i,4],i=1,13)	/1,2,3,4,5,6,7,8,9,10,11,12,
	UNDEF/,
(attrtbl[i,5],i=1,4)	/1, 2, 3, 4/,
(attrtbl[i,6],i=1,7)	/1b,2b,4b,10b,20b,40b,100b/

```
# -----
```

```
# Text to identify category in error messages
```

data (sname[i,1] ,i=1,2)	//movement direction//,
(sname[i,2] ,i=1,2)	//cycle interruption//,
(sname[i,3] ,i=1,2)	//sling/lifting device//,
(sname[i,4] ,i=1,2)	//operator identifier//,
(sname[i,5] ,i=1,2)	//cycle type //, # needs trlmg. blank
(sname[i,6] ,i=1,2)	//weight category//,

1 2 3 4 5 6 7 8 9

```

      (sname[i,7] ,i=1,2)      //lighter sequence//,
      (sname[i,8] ,i=1,2)      //special transporter//,
      (sname[i,9] ,i=1,2)      //cycle start lower//,
      (sname[i,10],i=1,2)      //cycle start upper//,
      (sname[i,11],i=1,2)      //cycle duration lower//,
      (sname[i,12],i=1,2)      //cycle duration upper//

```

```

*-----
rewind NSCFIL          # prepare to input user's selection criteria
rewind NSCINTR
while(.true.) {
  read(NSCFIL,80) dm, ci, sl, oi, ct, wc, ls, sx, cs1, csu,
                 cdl, cdu, nbucket, bktsize, prthist,
                 title, nscrec
  if (endfile(NSCFIL)) {
    rewind NSCINTR
    return
  }
  write (NREPORT,85) dm, ci, sl, oi, ct, wc, ls, sx, cs1, csu,
                 cdl, cdu, nbucket, bktsize, prthist,
                 title, nscrec
  call bcreset          # initialize for a new set of BY codes
*-----
  if (bycode(dm, 1, bcstat) == NO) {      # direction-of-movement
    if (dm == BLANK)
      selcrit[1] = ANY
    else if (dm == 1ro)
      selcrit[1] = OFFLOAD
    else if (dm == 1rr)
      selcrit[1] = RETROGD
    else {
      write (NERRRPT,2) (sname[i,1],i=1,2), nscrec
      next
    }
  } else if (bcstat == ERR) {
    write (NERRRPT,4) (sname[i,1],i=1,2), nscrec
    next
  }
*-----
  if (bycode(ci, 2, bcstat) == NO) {      # cycle interruption code
    if (ci == BLANK)
      selcrit[2] = ANY
    else if (ci == 1ru)
      selcrit[2] = UNINTER
    else if (ci == 1ri)
      selcrit[2] = INTER
    else {
      write (NERRRPT,2) (sname[i,2],i=1,2), nscrec
      next
    }
  } else if (bcstat == ERR) {

```

1 2 3 4 5 6 7 8 9

```
write (NERRRPT,4) (sname[i,2],i=1,2), nscrec
next
```

}

```
#-----
if (bycode(sl, 3, bcstat) == NO) {      # sling/lifting device
  if (sl == BLANK)
    selcrit[3] = ANY
  else if (sl == 1rh)
    selcrit[3] = 'hosa'
  else if (sl == 1rm)
    selcrit[3] = 'manl'
  else if (sl == 1ru)
    selcrit[3] = 'u'
  else {
    write (NERRRPT,2) (sname[i,3],i=1,2), nscrec
    next
  }
} else if (bcstat == ERR) {
  write (NERRRPT,4) (sname[i,3],i=1,2), nscrec
  next
}
#-----
```

```
if (bycode(oi, 4, bcstat) == NO) {      # operator ID
  if (oi == BLANK)
    selcrit[4] = ANY
  else {
    if (oi > 1rl) {                      # valid ID'S are A-L
      write (NERRRPT,2) (sname[i,4],i=1,2), nscrec
      next
    }
    selcrit[4] = oi
  }
} else if (bcstat == ERR) {
  write (NERRRPT,4) (sname[i,4],i=1,2), nscrec
  next
}
#-----
```

```
if (bycode(ct, 5, bcstat) == NO) {      # cycle type
  if (ct == BLANK)
    selcrit[5] = ANY
  else {
    if (ct > 1rd) {                      # valid codes are A-D
      write (NERRRPT,4) (sname[i,5],i=1,2), nscrec
      next
    }
    selcrit[5] = ct
  }
} else if (bcstat == ERR) {
  write (NERRRPT,2) (sname[i,8],i=1,2), nscrec
  next
}
```

1 2 3 4 5 6 7 8 9

}

```
if (bycode(wc[1], 6, bcstat) == NO) { # weight category
  if (wc[1] == BLANK)
    selcrit[6] = ANY
  else
    if (wstmskr(wc, selcrit[6])) # create weight category mask
      continue                  # if wc contains valid code(s)
    else {
      write (NERRRPT,2) (sname[i,6],i=1,2), nscrec
      next
    }
} else if (bcstat == ERR) {
  write (NERRRPT,4) (sname[i,6],i=1,2), nscrec
  next
}
```

```
if (ls == 1rb) # lighter sequence code
  selcrit[7] = BETWEEN
else if (ls == 1rw)
  selcrit[7] = WITHIN
else if (ls == BLANK)
  selcrit[7] = ANY
else if (ls >= 1r1 & ls <= 1r9) # single digit specifies
  selcrit[7] = ls - 1r0          # particular ltr. cycle
else {
  write (NERRRPT,2) (sname[i,7],i=1,2), nscrec
  next
}
```

```
if (sx == BLANK) # special transporter
  selcrit[8] = ANY
else if (sx == 1rl)
  selcrit[8] = LSHB
else if (sx == 1rs)
  selcrit[8] = SBBG
else if (sx == 1rx)
  selcrit[8] = EXCLUDE
else {
  write (NERRRPT,2) (sname[i,8],i=1,2), nscrec
  next
}
```

```
if (csl == 0) # cycle start lower-limit
  selcrit[9] = UNDEF
else
  selcrit[9] = minutes(csl)
```

```
if (csu == 0) # cycle start upper-limit
  selcrit[10] = UNDEF
```

1 2 3 4 5 6 7 8 9

```

      else
        selcrit[10] = minutes(csu)
#-----
      if (cdl == 0)                                     # cycle duration lower-limit
        selcrit[11] = UNDEF
      else
        selcrit[11] = minutes(cdl)
#-----
      if (cdu == 0)                                     # cycle duration upper-limit
        selcrit[12] = UNDEF
      else
        selcrit[12] = minutes(cdu)
#-----
# IF BY-code table is properly set up, generate individual SC
# records for each item selected for BY-explosion. See also BYCODE
# for a better understanding of the BY-generation table.
#-----
      if (bstblok(dummy)) {

        if (bysentb[1] ~= 0) {      # perform BY code "explosion"
          b1l = atlimit[bysentb[1]]
          do b1 = 1, b1l
          {
            selcrit[bysentb[1]] = attrtbl[b1,bysentb[1]]
            if (bysentb[2] ~= 0) {
              b2l = atlimit[bysentb[2]]
              do b2 = 1, b2l
              {
                selcrit[bysentb[2]] = attrtbl[b2,bysentb[2]]
                if (bysentb[3] ~= 0) {
                  b3l = atlimit[bysentb[3]]
                  do b3 = 1, b3l
                  {
                    selcrit[bysentb[3]] = attrtbl[b3,bysentb[3]]
                    if (bysentb[4] ~= 0) {
                      b4l = atlimit[bysentb[4]]
                      do b4 = 1, b4l
                      {
                        selcrit[bysentb[4]] = attrtbl[b4,bysentb[4]]
                        if (bysentb[5] ~= 0) {
                          b5l = atlimit[bysentb[5]]
                          do b5 = 1, b5l
                          {
                            selcrit[bysentb[5]] = attrtbl[b5,bysentb[5]]
                            if (bysentb[6] ~= 0) {
                              b6l = atlimit[bysentb[6]]
                              do b6 = 1, b6l
                              write (NSCINTR) selcrit, auxinfo
                                selcrit[bysentb[6]] = attrtbl[b6,bysentb[6]]
                              } else

```

1 2 3 4 5 6 7 8 9

```

        write (NSCINTR) selcrit, auxinfo
    }
    } else
        write (NSCINTR) selcrit, auxinfo
    }
    } else
        write (NSCINTR) selcrit, auxinfo
    }
    } else
        write (NSCINTR) selcrit, auxinfo
    }
    } else
        write (NSCINTR) selcrit, auxinfo
    }
    } else
        write (NERRRPT,90) nscrec # error in BY-code specification
    } # end of "repeat"
}
-----
2 format('0error in selection code for category ',2a10,
' in record no. ',i6)

4 format('0error in by code for selection of ',2a10,
' in record no. ',i6)

80 format(5(r1,1x), # dm, ci, sl, oi, ct
6r1,1x, # wc
2(r1,1x), # ls, sx
4i4, # csl, csu, cdl, cdu
i3,1x, # nbucket
f6.2,1x, # bktsize
r1,1x, # prthist
4a10, # title
i6 ) # nscrec

85 format('/' selection record-- ',
5(r1,1x), # dm, ci, sl, oi, ct
6r1,1x, # wc
2(r1,1x), # ls, sx
4(i4,1x), # csl, csu, cdl, cdu
i3,1x, # nbucket
f6.2,1x, # bktsize
r1,1x, # prthist
4a10, # title
i6 ) # nscrec

90 format('0error in by-code specification in sc record no. ',i6)

end

```

1 2 3 4 5 6 7 8 9

```
integer function minutes (hhmm) # convert hrs./mins. code to minutes
integer hours, hhmm
hours = hhmm/100
minutes = hours * 60 + hhmm - hours * 100
end
```


1 2 3 4 5 6 7 8 9

```
# Test SCFIELD for being a "BY" code & set up BY-generation table
```

```
#-----
# If the code passed as the value of the SCFIELD parameter is
# in the range of valid BY-codes, and this code has not already been
# specified on the same selection criteria record (since the last
# time BCRESET was called), save the value of the SCNUM parameter in
# the element of the BYGENTB array specified by SCFIELD and return
# YES as the function value, as well as through the RETCODE parameter
#
```

```
# The BY-generation table will be used by the routine CRSCINT
# (which is the sole caller of BYCODE) to perform the selection
# criteria "explosion" function (generation of multiple SC records
# for records from SCFILE specifying "BY" selections.)
#-----
```

```
integer function bycode (scfield, scnum, retcode)
```

```
implicit integer (a-z)
common /bst/ bysentb[6]
```

```
#-----
if (scfield > 1r0 & scfield <= 1r6) { # if valid BY ordering num.
    bynum = scfield - 1r0                # (convert to integer)
    if (bysentb[bynum] == 0) {           # and num. not previously used
        bysentb[bynum] = scnum          # set proper element of BY-generation
        bycode = retcode = YES          # table to sel. crit. category number
    } else
        bycode = retcode = ERR          # user selected same BY number
                                         # more than once
    } else
        bycode = retcode = NO           # not a valid BY code

return
```

```
#-----
entry bcreset      # initialize BYCODE array for next SC set
```

```
bysentb[1]=bysentb[2]=bysentb[3]=bysentb[4]=bysentb[5]=bysentb[6] = 0
```

```
end
```

1 2 3 4 5 6 7 8 9

* See if BY-generation table is set up properly

*-----

logical function bstblok (dummy)

implicit integer (a-z)

logical zrfound

* zero-element-found flag

common /bst/ bysentb[6]

*-----

```

if (bysentb[1] == 0) {
    do i= 2, 6
    {
        if (bysentb[i] /= 0) {
            bstblok = .false.
            return
        }
    }
    bstblok = .true.
} else {
    zrfound = .false.
    do i= 1, 6
    {
        if (zrfound) {
            if (bysentb[i] /= 0) {
                bstblok = .false.
                return
            }
        } else
            if (bysentb[i] == 0)
                zrfound = .true.
    }
    bstblok = .true.
}

end

```

* if 1st element is zero,
* check that all others are also

* ERROR--some element is non-zero

* all elements are zero

* search for a zero element

* all non-zero elements are compact

1 2 3 4 5 6 7 8 9

```
# create weight mask from individual category selection codes
```

```
#-----
# logical function wgtmskr (wstcode, rtnmask)
# implicit integer (a-z)
# integer wstcode[6]
```

```
#-----
# Accepts 6-element array of right-justified Hollerith-
# encoded characters. For each character in the range
# A-F, sets appropriate bit in RTNMASK.
# (A= bit 0, B= bit 1, . . . , F= bit 5)
# For letter 'N', set bit 7.
# (used to indicate selection of all cycles for containers
# with no weight code)
```

```
#-----
# N.B. If any of the following code looks obscure, here's a hint--
```

```
# 1ra = 1 (rt.-justified letter = decimal integer)
# 1rb = 2
# .
# .
# .
# 1rz = 26
```

sst

```
#-----
bitter(nbit) = shift(1,nbit-1) # (statement function)
```

```
wgtmskr = .true.
rtnmask = 0
```

```
# for (i=1; i<=6; i=i+1) {
#   i = 1
#   while (i <= 6) {
#     if (wstcode[i] <= 1rf)      # select wt. category A-F
#       rtnmask = rtnmask | bitter(wstcode[i])
#     else if (wstcode[i] == 1rn) # select records with no wt. code
#       rtnmask = rtnmask | 100b
#     i = i + 1
#   }

#   if (rtnmask == 0)            # if no bits were set,
#     wgtmskr = .false.         # no valid codes were selected

end
```

1 2 3 4 5 6 7 8 9

overlay('cyclovr',3,0)

Program reportr # supervise gen. of cycle data statistics report

```
#-----
#       For each "group" within the DBINTR file, range through each
#       set of selection criteria contained in the SCINTR file, selecting
#       records within that group whose attributes cause them to qualify,
#       saving the cycle times associated with this set, and passing them
#       to WRITRPT, along with appropriate information needed in order to
#       produce a report containing statistics for the particular set of
#       records selected.
```

```
#       After all sets of selection criteria have been applied to a
#       particular group, call FACSUM to produce a summary report
#       containing statistics which apply to the group as a whole
#       (i.e., the set of all selected sets).
```

```
#-----
implicit integer (a-z)
logical tblmakr, setslcr, select
integer selcrit[NSC]           # SC codes for SELECT
integer scwords[NSCW]          # Hollerith codes for WRITRPT
integer brkkeys[3]             # break keys for WRITRPT
integer title[4]               # report headings
integer rctimes[MAXCYCLE]      # retrieved cycle times
integer ncyrctr                # no. of cycle times retrieved
integer bktsize                # size of each histogram "bucket"
#-----
call sswtch(1,sw1)             # dump facility table to NERRRPT
call sswtch(2,sw2)             # produce statistics report

while (tblmakr(brkkeys[1])) {   # create cycle/attribute table
                                #   for next facility group
                                #   if another group exists

    if (sw2 == 1)               # if option selected,
        call tblmcr(brkkeys[1]) #   dump facility table

    while (getslcr(selcrit,      # for next set of selection criteria
                    title, bktsize, #   (if one exists),
                    nbucket, rthist)) {

        while (select(selcrit,   # return indices of facility table
                        rctimes,  # cycles satisfying these criteria
                        brkkeys[2], brkkeys[3])) { # (while cycle groups remain)

            call mkscwrds(selcrit, scwords) # gen. SC titles

            if (ncyrctr > 0) {

                #####
                # replace facility table indices from GETSLCR #
                # with cycle times in RCTIMES array and #
```

1 2 3 4 5 6 7 8 9

```
      #      print selected Cargo Cycle Data      #
      #      if switch 3 is set.                  #
      #####
      call cvtcccd(rctimes, ncyrctr, title, scwords,
                   brkkeys, prthist, bktsize, nbucket)

      if (sw1 == 1)
        call writrct(rctimes, ncyrctr, title, scwords,
                    brkkeys, prthist, bktsize, nbucket)
      }
    }
    if (sw1 == 1)
      # produce summary report
      #   for entire facility
      call facsum(rctimes, ncyrctr, title, scwords,
                  brkkeys, prthist, bktsize, nbucket)
    }
  }
end
```

1 2 3 4 5 6 7 8 9

logical function tblmakr (facname) # construct facility table

implicit integer (a-z)

logical fstcall, eofflag, endfile

common

dsilast,

opdate[NFTSEG], opshft[NFTSEG], ftbindx[NFTSEG],

dirmvmt[FTSIZE], cwtpe[FTSIZE], ssifdev[FTSIZE],

operid[FTSIZE], wshcat[FTSIZE], cystart[FTSIZE],

cydurat[FTSIZE], spclxpt[FTSIZE], lstrseq[FTSIZE],

cwinter[FTSIZE]

integer padding[6]

data fstcall /.true./, eofflag/.false./, padding/6*0/

```

#-----
if (eofflag) {                                # if EOF on DBINTR on previous
    tblmakr = .false.                          # entry, return EOF signal
    return
}

```

```

#-----
if (fstcall) {                                # read file header on first call

    rewind NDBINTR
    read (NDBINTR) filename, dbname, fcdte, fctime, dbdate, dbtime
    if (endfile(NDBINTR)) {
        write (NERRRPT,10)                    # d. b. intermediate file is null
        call remark('null db intermediate file')
        stop
    }
    if (filename /= 'dbintr') {
        write (NERRRPT,20) filename            # wrong file used
        call remark('wrong file on unit NDBINTR detected by tblmakr')
        stop
    }
    write (NREPORT,5) fcdte, fctime, dbdate, dbtime, dbname
    fstcall = .false.
}

```

```

#-----
                                # read first group header
                                # (directly into OPDATE/OPSHFT arrays)
read (NDBINTR) bosflag, facname, opdate[1], opshft[1], padding
if (bosflag /= BOG) {
    write (NERRRPT,30)
    call remark('header error in db intermediate file')
    stop
}

```

```

#-----
ftbindx[1] = 1                                # set pointer to first table segment
ftx = 1                                       # initialize facility table index
ptx = 2                                       # and pointer array index

```

1 2 3 4 5 6 7 8 9

```
read (NDBINTR) w1, w2, w3, w4, w5, w6, w7, w8, w9, w10
```

```
while (~endfile(NDBINTR)) {
```

```
    if (w1 == BOG) {                                # NEW GROUP HEADER PROCESSING
        odate[ftx] = w3                               # enter opn. date & shift code
        opshft[ftx] = w4                             # into index arrays
        ftbindx[ftx] = ftx                           # set index pointer
        ftx = ftx + 1                                # to next available position
        if (w2 ~= facname) {                          # if new group is for new facility,
            backspace NDBINTR                         # reread header on next entry
            tblmakr = .true.
            dsilast = ftx - 1                          # set pointer to end of index list
            return
        }
    }
    else { # not new group hdr--save record info into facility table
        if (ftx > FTSIZE) {
            call remark('facility table overflow in tblmakr')
            write (NERRRPT,40)
            stop
        }
        dirmvmt[ftx] = w1;    wshtcat[ftx] = w5;    spclxpt[ftx] = w8
        cytype[ftx] = w2;    cstart[ftx] = w6;    lstrseq[ftx] = w9
        operid[ftx] = w3;    cxdurat[ftx] = w7;    cwinter[ftx] = w10
        sslfdev[ftx] = w4
        ftx = ftx + 1
    }
    read (NDBINTR) w1, w2, w3, w4, w5, w6, w7, w8, w9, w10
}
```

```
ftbindx[ftx] = ftx                                #END-OF-FILE processing
dsilast = ftx                                     # set last index pointer to end
eoflas = tblmakr = .true.                        # of final table segment + 1
return
```

```
5 format('Ousing lots intermediate data file created on',a9,
' at',a9,' from raw data file of',a9,' --',a9,
// raw data file name is ',a10)
```

```
10 format('Odata base intermediate file is null')
```

```
20 format('Owrong file read on unit NDBINTR--',
'file name is ',a6)
```

```
30 format('Otblmakr detected unreadable group header')
```

10/16/78

13.48.54.

TBLMAKR

PAGE NO. 3

1 2 3 4 5 6 7 8 9

40 format('Ofacility table overflow--current limit is FTSIZE')

end

1 2 3 4 5 6 7 8 9

subroutine tblmper (facname) # dump facility table

implicit integer (a-z)

common dsilast,
 opdate[NFTSEG], opshft[NFTSEG], ftbindx[NFTSEG],
 dirmvmt[FTSIZE], cytype[FTSIZE], sglfdev[FTSIZE],
 operid[FTSIZE], wshtcat[FTSIZE], cystart[FTSIZE],
 cydurat[FTSIZE], spclxpt[FTSIZE], lstrseq[FTSIZE],
 cyinter[FTSIZE]

```
write (NERRRPT,10)
i = 1
while (i < dsilast) {
  write (NERRRPT,20) facname, opdate[i], opshft[i]
  j = ftbindx[i]
  last = ftbindx[i+1] - 1
  while (j <= last) {
    wtcodes = wcident(wshtcat[j])
    write (NERRRPT,30) j,
      dirmvmt[j], cyinter[j], sglfdev[j], operid[j],
      cytype[j], wtcodes, lstrseq[j], spclxpt[j],
      cystart[j], cydurat[j]

    j = j + 1
  }
  i = i + 1
}
```

10 format(//1x,46('*'),' facility table dump ',46('*')//)

20 format(// facility date shift//1x,3a10/6x,
 'movement'3x'interrupt'2x'sl/lf dev'2x'operator'3x,
 'cycle type'1x'weight cat'1x'ltr seq'4x'spcl xpt'3x,
 'cyclestart'1x'cycledurat//)

30 format(1x,i4, # cycle number
 1x,a10, # direction-of-movement
 1x,a10, # cycle interruption
 1x,a10, # s/l device
 1x,i10, # operator ID
 1x,i10, # cycle type
 1x,a10, # weight category
 1x,i10, # lighter sequence
 1x,a10, # special transporter
 2(1x,i10)) # cystart, cydurat
 end

1 2 3 4 5 6 7 8 9

```
# Get selection criteria from intermediate file created by CRSCINT
# returns .FALSE. on EOF, .TRUE. otherwise
```

```
-----
logical function getslcr (selcrit, title, bktsize, nbucket, prthist)
```

```
implicit integer (a-z)
integer selcrit[12], title[4]
logical endfile
common /nscrec/ nscrec
```

```
-----
setslcr = .true.
read (NSCINTR) selcrit, nbucket, bktsize, prthist, nscrec, title
```

```
if (endfile(NSCINTR)) {
    setslcr = .false.
    rewind NSCINTR
}
```

```
end
```

1 2 3 4 5 6 7 8 9

```
# Select and return a set of cycle times
```

```
#-----
# Scan current facility table (most recently constructed by
# TBLMAKR) for those cycles which satisfy the criteria given in
# the SC array passed by the caller. Return the cycle times indices
# from all such cycles in the CYCLES array (through the formal
# parameter list), and set the FP NCYCLES to the value of the
# number of cycles being returned.
```

```
# Set the variables SDATE and SSHIFT to indicate the date and
# shift of the operation from which the cycles were obtained.
```

```
#-----
logical function select(sc, cycles, ncycles, sdate, sshift)
```

```
implicit integer (a-z)
integer sc[12] # selection criteria array
integer cycles[MAXCYCLE] # cycle times returned to caller
integer ncycles # no. of cycles being returned
integer sdate # selected group date
integer sshift # selected group shift
integer dsindex # index for date/shift/pointer table
common /nscrec/ nscrec
common dsilast,
opdate[NFTSEG], opshift[NFTSEG], ftbindx[NFTSEG],
diravmt[FTSIZE], ctype[FTSIZE], selfdev[FTSIZE],
operid[FTSIZE], wshtcat[FTSIZE], cystart[FTSIZE],
cydurat[FTSIZE], spolxpt[FTSIZE], lstrseg[FTSIZE],
cwinter[FTSIZE]
```

```
data dsindex/0/ # *** FOR FIRST CALL ONLY ***
```

```
#-----
ncycles = 0 # initialize cycle counter
dsindex = dsindex + 1 # and pointer table index
sdate = opdate[dsindex] # return selected group date
sshift = opshift[dsindex] # and shift
if (dsindex >= dsilast) { # check for end of current data table
  select = .false.
  dsindex = 0 # re-initialize pointer for next call
  return
}
# for (i=ftbindx[dsindex]; i<ftbindx[dsindex+1]; i=i+1) {
i = ftbindx[dsindex]
while(i < ftbindx[dsindex+1]) {

  if (ncycles >= MAXCYCLE) {
    write(NERRRPT,10) sdate, sshift, nscrec
    call remark('too many cycles selected')
    stop
  }
  if (diravmt[i] == sc[1] | sc[1] == ANY)
  if (cwinter[i] == sc[2] | sc[2] == ANY)
```

1 2 3 4 5 6 7 8 9

```

if (sslfdev[i] == sc[3] | sc[3] == ANY)
if (operid[i] == sc[4] | sc[4] == ANY)
if (cotype[i] == sc[5] | sc[5] == ANY)
if ((wshtcat[i] & sc[6]) ~= 0 | sc[6] == ANY) # test wt.-code bit
if ((sc[7] == ANY) # all cycles
    |(sc[7] == BETWEEN & lstrseq[i] == 0) # inter-cycle
    |(sc[7] == WITHIN & lstrseq[i] ~= 0) # intra-cycle
    |(sc[7] >= 0 & lstrseq[i] == sc[7])) # specific cycle

```

```

if ((sc[8] == ANY) # all transporters
    |(sc[8] == LSHB & spclxpt[i] == LSHB) # lash barge
    |(sc[8] == SBBG & spclxpt[i] == SBBG) # seabee barge
    |(sc[8] == EXCLUDE & spclxpt[i] ~= LSHB # neither lash
        & spclxpt[i] ~= SBBG)) # nor seabee barge

```

```

if ((sc[9]==UNDEF | cystart[i]>=sc[9]) # cycle start time
    & (sc[10]==UNDEF | cystart[i]<=sc[10])) # must be within

```

```

if ((sc[11]==UNDEF | cydurat[i]>=sc[11]) # limits unless
    & (sc[12]==UNDEF | cydurat[i]<=sc[12])) { # limits not given

```

```

    ncycles = ncycles + 1
    cycles[ncycles] = i # return cycle index

```

```

}

```

```

i = i + 1

```

```

}

```

```

select = .true. # not yet end of SC data

```

```

10 format('0cycle overflow in selection for date ',a3,
    ' shift ',a1/' current size of cycle table is MAXCYCLE',
    ' selection criteria record no. ',i6)

```

```

end

```

1 2 3 4 5 6 7 8 9

* Fabricate selection criteria title words for WRITRPT.

```

*-----
subroutine mkscwrđ (sccode, scword)

implicit integer (a-z)
integer sccode[NSC], scword[NSCW], temp[4]
*-----
scword[1] = sccode[1]           # dir-of-mvmt
scword[2] = sccode[2]           # cycle interruption
scword[3] = sccode[3]           # slings/lift device
*-----
if (sccode[4] >= 1ra & sccode[4] <= 1rl) # operator ID
    encode (8, 15, scword[4]) sccode[4]
else
    scword[4] = sccode[4]
*-----
if (sccode[5] >= 1ra & sccode[5] <= 1rd) # cycle type
    encode (10, 40, scword[5]) sccode[5]
else
    scword[5] = sccode[5]
*-----
if (sccode[6] == ANY)           # wt. category
    scword[6] = ANY
else
    scword[6] = wcident(sccode[6])
*-----
if (sccode[7] >= 1 & sccode[7] <= 9) # lighter sequence
    encode (10, 10, scword[7]) sccode[7]
else
    scword[7] = sccode[7]
*-----
scword[8] = sccode[8]           # special xporter
*-----
do i= 1, 4                      # cycle start/dur.
    if (sccode[i+8] == UNDEF)
        temp[i] = '....'
    else
        encode (4, 20, temp[i]) sccode[i+8]
*-----
encode (10, 30, scword[9]) temp[1], temp[2]
encode (10, 30, scword[10]) temp[3], temp[4]
*-----

10 format('sequence ',i1)
15 format('oper. ',i2)
20 format(i4)
30 format(a4,'-',a4)
40 format('cycle tr ',r1)

end

```

1 2 3 4 5 6 7 8 9

Encode a 10-character word with a printable weight code
designator, based on bits set in input parameter value WTCODE.

integer function wcident (wtcode)

implicit integer (a-z)
integer codes[7]

do i= 1, 6

if (and(shift(wtcode, 1-i), 1) /= 0)

codes[i] = i # cf. "N.B." comment in WGTMSKR

else

codes[i] = BLANK

if (and(shift(wtcode, -6), 1) /= 0)

codes[7] = 1rn

else

codes[7] = BLANK

encode (10, 10, wcident) codes

10 format('wt.', 7r1)

end

1 2 3 4 5 6 7 8 9

```
# Convert cycles and Print selected Cargo Cycle Data.
```

```
#-----
# 1) Convert values in RCTIMES array from indices of selected
# facility table elements to values of cycle times for those indices.
#
# 2) If option selected by settings of sense switch 3, print
# 'exploded' selection criteria values from SCWORDS, and
# print values of facility table cycle data actually selected.
#-----
```

```
subroutine cvtpccd (rctimes, ncycrtr, title, scwords,
                  brkkeys, prthist, bktsize, nbucket)
implicit integer (a-z)
integer rctimes(ncycrtr), title[4], scwords[NSCW], brkkeys[3]
common
    dsilast,
    odate[NFTSEG], opshft[NFTSEG], ftbindx[NFTSEG],
    dirmvmt[FTSIZE], ctype[FTSIZE], sselfdev[FTSIZE],
    operid[FTSIZE], wshtcat[FTSIZE], cystart[FTSIZE],
    cydurat[FTSIZE], spclxpt[FTSIZE], lstrseq[FTSIZE],
    cyinter[FTSIZE]
```

```
#-----
call sswtch(3,switch)          # print table info. if sw. 3 set
if (switch == 1)
    write (NREPORT, 10) title, prthist, bktsize, nbucket, scwords,
        brkkeys, ncycrtr
if (ncycrtr /= 0)
    do i = 1, ncycrtr
    {
        j = rctimes[i]          # replace index with cycle value
        rctimes[i] = cydurat[j] # for routine WRITRPT
        if (switch == 1) {
            wtcodes = wcident(wshtcat[j])
            write (NREPORT, 20) j,
                dirmvmt[j], cyinter[j], sselfdev[j], operid[j],
                ctype[j], wtcodes, lstrseq[j], spclxpt[j],
                cystart[j], cydurat[j]
        }
    }
    }
return
```

```
#-----
10 format(/1x, '*** ', 4a10, 15x, 'histogram=', r1, 5x, 'bktsize= ', f6.2,
    5x, 'no. of buckets= ', i3/6x,
    'movement' 3x 'interrupt' 2x 's1/lf dev' 2x 'operator' 3x,
    'cycle type' 1x 'weight cat' 1x 'ltr seq' 4x 'spcl xpt' 3x,
    'cyclestart' 1x 'cycledurat' /5x, NSCW a11 /6x,
    'facility date      shift' /6x, 3a10/1x, i4, ' cycles selected')
```

```
20 format(1x, i4,          # cycle number
    1x, a10,              # direction-of-movement
    1x, a10,              # cycle interruption
    1x, a10,              # s/l device
```

1 2 3 4 5 6 7 8 9

1x,i10, # operator ID

1x,i10, # cycle type

1x,a10, # weight category

1x,i10, # lighter sequence

1x,a10, # special transporter

2(1x,i10) # cstart, cdurat

30 format('0',i5,' total cycles selected for facility')

end


```

SUBROUTINE WRITRPT (CYTIM,NOTIMS,TITLE,SELCRI,BRKEYS,HISTSW,
1      BINSIZ,NOBIN)
C      PURPOSE- TO CALL STATS TO COMPUTE STATISTICS (MEAN,STANDARD
C      DEIVATION, AND OPTIONALLY A MEDIAN AND HISTOGRAM)
C      BASED ON A SET OF INPUT CYCLE TIMES AND, WHEN APPROPRIATE
C      (I.E. AT EACH FACILITY BREAK), SHIFT-ACCUMULATED
C      STATISTICS. THEN TO PRINT THESE OUT WITH APPROPRIATE
C      HEADINGS. THIS ROUTINE IS CALLED EACH TIME THERE IS A
C      BREAK ON FACILITY, DATE OR SHIFT. AN ENTRY POINT IN THIS
C      ROUTINE IS USED TO PRINT SHIFT-ACCUMULATED STATISTICS
C      WHENEVER FACILITY CHANGES.

      REAL SELCRI,BINSIZ,XBAR,SDEV,MEDIAN,BINFREQ,
1      BINCUM
      INTEGER NOBIN,SHIFT,NOTIMS,SNOTIMS,TITLE(4),SELCRI(10),CYTIM(200)
      DIMENSION BRKEYS(3),BINFREQ(100),
1      BINREL(100),BINCUM(100),SBINREL(100,2), SBINCUM(100,?),
2      SHMEAN(2),SHSDEV(2),SMEDIAN(2)
      COMMON/SHFSTAT/SHSUM(2),SHSUMSQ(2),SHBINFR(100,2),SNOTIMS(2)
      DATA BLANK/10H /
      DATA SMEDIAN/2*0./
      DATA SBINCUM/200*0./
      DATA SBINREL/200*0./
C      SHFSTAT TRANSPORTS SHIFT ACCUMULATED STATISTICS NEEDED WHEN
C      ENTRY POINT ENTRY OCCURS.
      DECODE(10,190,BRKEYS(3)) SHIFT
190  FORMAT( I1)
C      WRITE PAGE HEADER
      WRITE(6,100) TITLE
100  FORMAT(1H1,20X,4A10 / )
      WRITE(6,110) BRKEYS(1),BRKEYS(2),SHIFT
110  FORMAT(26X,13HFACILITY ID- A4, 2X, 6HDATE- A3,5X,7HSHIFT- I2
1  /10X,19HSELECTION CRITERIA- )
C      PRINT NONBLANK VALUED SELECTION CRITERION ONLY
      IF( SELCRI(1) .NE. BLANK ) WRITE(6,111) SELCRI(1)
111  FORMAT( 38X,22HDIRECTION OF MOVEMENT- A10 )
      IF( SELCRI(2) .NE. BLANK ) WRITE(6,112) SELCRI(2)
112  FORMAT( 42X,18HINTERRUPTION CODE- A10 )
      IF( SELCRI(3) .NE. BLANK ) WRITE(6,113) SELCRI(3)
113  FORMAT( 34X,26HSLING/LIFTING DEVICE TYPE- A10 )
      IF( SELCRI(4) .NE. BLANK ) WRITE(6,115) SELCRI(4)
115  FORMAT( 48X,12HOPERATOR ID- A10 )
      IF( SELCRI(5) .NE. BLANK ) WRITE(6,116) SELCRI(5)
116  FORMAT( 49X,11HCYCLE TYPE- A10 )
      IF( SELCRI(6) .NE. BLANK ) WRITE(6,114) SELCRI(6)
114  FORMAT( 44X,16HWEIGHT CATEGORY- A10 )
      IF(SELCRI(7) .NE. BLANK ) WRITE(6,120) SELCRI(7)
120  FORMAT( 39X,21HINTRA/INTER CARRIER - A10 )
      IF( SELCRI(8) .NE. BLANK ) WRITE(6,117) SELCRI(8)
117  FORMAT( 35X,25HSPECIAL TRANSPORTER CODE- A10 )
      IF( SELCRI(9) .NE. BLANK ) WRITE(6,118) SELCRI(9)

```

```

118 FORMAT( 36X,24HCYCLE START TIME PERIOD- A10 )
    IF( SELCRI(10) .NE. BLANK ) WRITE(6,119) SELCRI(10)
119 FORMAT (38X,22HCYCLE DURATION LIMITS- A10 )
C   CALL SUBROUTINE STATS TO COMPUTE MEAN,MEDIAN,STANDARD DEVIATION
C   AND HISTOGRAM VALUES(OPTIONAL) BASED ON A SET OF CYCLE TIMES
C   (CYTIM) AND ACCUMULATE SHIFT ASSOCIATED STATISTICS.
    CALL STATS( CYTIM,NOTIMS,HISTSW,BINSIZ,NOBIN,XBAR,SDEV,MEDIAN,
1       BINFREQ,BINREL,BINCUM,SHIFT)
C   PRINT FACILITY/DATE/SHIFT BREAK REPORT
    WRITE(6,130) NOTIMS, (CYTIM(I),I=1,NOTIMS)
130 FORMAT( 3X, I4,18H CYCLE TIMES(MIN)- 2X, 10I10/(27X,10I10) )
    WRITE(6,140) XBAR,SDEV
140 FORMAT( 10X, 6HMEAN- F10.2,10X,14HSTANDARD DEV- F10.2 )
    IF(HISTSW .NE. 1RH) GO TO 1000
    WRITE(6,150) MEDIAN,NOBIN,BINSIZ
150 FORMAT( 8X,8HMEDIAN- F10.1,/ 10X,19HHISTOGRAM VALUES- I4, 1X,
1       13HBINS OF SIZE F10.2 / 10X, 3HBIN 7X, 9HFREQUENCY
2       6X, 13HRELATIVE FREQ 2X,12HCUM REL FREQ )
    DO 15 I=1,NOBIN
        BIN1=(I-1)*BINSIZ
        BIN2=BIN1 + BINSIZ
        WRITE(6,160)BIN1,BIN2,BINFREQ(I),BINREL(I),BINCUM(I)
160 FORMAT( 2X, F6.1, 3H - F6.1,2X, F7.1, 6X, F7.1, 6X, F7.1 )
    15 CONTINUE
1000 CONTINUE
    RETURN
    ENTRY FACSUM
    IF(SNOTIMS(1) .EQ. 0 .AND. SNOTIMS(2) .EQ. 0) GO TO 56
C   WRITE PAGE HEADER
    WRITE(6,100) TITLE
    WRITE(6,200) BRKEYS(1)
200 FORMAT( 10X,32HSHIFT STATISTICS FOR FACILITY - A4 / )
C   COMPUTE SHIFT STATISTICS
    DO 25 I=1,2
        IF(SNOTIMS(I) .EQ. 0) GO TO 25
        SHMEAN(I)=SHSUM(I)/SNOTIMS(I)
        SHSDEV(I)=SHSUMSQ(I) /SNOTIMS(I) -
1       SHMEAN(I) * SHMEAN(I)
        SHSDEV(I)=SQRT(SHSDEV(I))
        SMEDIAN(I)=0.
        DO 30 J=1,NOBIN
            SBINREL(J,I) = SHBINFR(J,I)/SNOTIMS(I)
30 CONTINUE
            SBINCUM(1,I) = SBINREL(1,I)
            DO 35 J=2,NOBIN
                SBINCUM(J,I) = SBINCUM(J-1,I) + SBINREL(J,I)
                IF( SBINCUM(J,I) .LT. .5 ) GO TO 35
                IF( SMEDIAN(I) .EQ. 0. ) SMEDIAN(I) = (J-0.5)*BINSIZ
35 CONTINUE
C   PRINT SHIFT STATISTICS

```

```

        WRITE(6,210) I,SNOTIMS(I)
210    FORMAT(/9X, 6HSHIFT I1, 11H - BASED ON I4, 12H CYCLE TIMES / )
        WRITE(6,140)SHMEAN(I),SHSDEV(I)
        IF(HISTSW .NE. 1RH) GO TO 48
        WRITE(6,150) SMEDIAN(I),NOBIN,BINSIZ
        DO 45 J=1,NOBIN
            BIN1=(J-1)*BINSIZ
            BIN2= BIN1 + BINSIZ
            WRITE(6,160) BIN1,BIN2,SHBINFR(J,I),SBINREL(J,I),
1          SBINCUM(J,I)
45    CONTINUE
C    INITIALIZE SHIFT ASSOCIATED PARAMETERS
48    CONTINUE
        SHSUM(I) = 0.
        SHSUMSQ(I) = 0.
        SNOTIMS(I) = 0
        SMEDIAN(I)=0.
        DO 50 J=1,NOBIN
            SHBINFR(J,I) = 0.
            SBINCUM(J,I)=0.
            SBINREL(J,I)=0.
50    CONTINUE
25    CONTINUE
56    CONTINUE
        RETURN
        END

```

```

SUBROUTINE STATS (CYTIM,NOTIMS,HISTSW,BINSIZ,NOBIN,XBAR,SDEV,
1      MEDIAN,BINFREQ,BINREL,BINCUM,SHIFT)
C  PURPOSE-  TO COMPUTE A MEAN,STANDARD DEVIATION-MEDIAN AND
C            (OPTIONALLY) HISTOGRAM VALUES BASED ON A SET OF INPUT
C            CYCLE TIMES. IN ADDITION, TO ACCUMULATE SHIFT ASSOCIATED
C            STATISTICS.
      DIMENSION BINFREQ(100),BINREL(100),BINCUM(100)
      INTEGER CYTIM(200)
      INTEGER SHIFT,SNOTIMS
      REAL MEDIAN
      COMMON/SHFSTAT/SHSUM(2),SHSUMSQ(2),SHBINFR(100,2),SNOTIMS(2)
C  INITIALIZE SHIFT ASSOCIATED PARAMETERS
      DATA SHSUM,SHSUMSQ,SNOTIMS/ 2*0.,2*0.,2*0/,
1      SHBINFR/200*0./
C  INITIALIZE NON SHIFT ASSOCIATED STATISTICS
      XBAR=0.
      SDEV=0.
      MEDIAN=0.
      DO 5 J=1,100
        BINFREQ(J)=0.
        BINREL(J)=0.
        BINCUM(J)=0.
5  CONTINUE
      SUM=0.
      SUMSQ=0.
C  MEAN AND STANDARD DEVIATION
      DO 10 I=1,NOTIMS
        CYC=CYTIM(I)
        SUM= SUM + CYC
        SUMSQ= SUMSQ + CYC*CYC
10  CONTINUE
      XBAR= SUM/NOTIMS
      SDEV= SUMSQ/NOTIMS - XBAR*XBAR
      SDEV=SQRT(SDEV)
C  HISTOGRAM VALUES (IF HISTSW = TRUE)
C  (NOTE-ENDPOINTS OF FIRST BIN - 0 TO 1*BINSIZ
C            ENDPOINTS OF JTH BIN  -(J-1)*BINSIZ TO J*BINSIZ
C            ENDPOINTS OF LAST BIN  -(NOBIN-1)*BINSIZ TO NOBIN*BINSIZ)
C  BIN FREQUENCY
      DO 15 I=1,NOTIMS
        CYC=CYTIM(I)
        DO 20 J=1,NOBIN
          K=NOBIN - J
          IF(CYC .LE. K*BINSIZ) GO TO 20
          BINFREQ(K+1)=BINFREQ(K+1) + 1
          GO TO 15
20  CONTINUE
15  CONTINUE
C  RELATIVE BIN FREQUENCY
      DO 25 J=1,NOBIN

```

```

        BINREL(J)=BINFREQ(J)/NOTIMS
25 CONTINUE
C      MEDIAN AND CUMULATIVE RELATIVE FREQUENCY
        BINCUM(1)=BINREL(1)
        DO 30 J=2,NOBIN
            BINCUM(J)= BINCUM(J-1) + BINREL(J)
C      FIND MAXIMUM BINCUM (ASSUMED NOT TO BE THE FIRST) WITH VALUE LESS
C      THAN .5 AND USE MIDPOINT AS APPROXIMATE MEDIAN
C      ( (J-2)*INTSIZ + .5*INTSIZ )
            IF ( BINCUM(J) .LT. .5 ) GO TO 30
            IF( MEDIAN .EQ. 0.) MEDIAN = (J-0.5)*BINSIZ
30 CONTINUE
C      ACCUMULATE SHIFT ASSOCIATED STATISTICS
        SHSUM(SHIFT)= SHSUM(SHIFT) + SUM
        SHSUMSQ(SHIFT)= SHSUMSQ(SHIFT) + SUMSQ
        SNOTIMS(SHIFT)= SNOTIMS(SHIFT) + NOTIMS
        DO 35 J=1,NOBIN
            SHBINFR(J,SHIFT)= SHBINFR(J,SHIFT) + BINFREQ(J)
35 CONTINUE
        RETURN
        END

```

1 2 3 4 5 6 7 8 9

```
logical function endfile (nunit)
if(eof(nunit) /= 0) {
    endfile = .true.
    return
}
endfile = .false.

end
```

NCARDF	1
NSCFIL	2
NDBINTR	3
NSCINTR	4
NREPORT	5
NERRRPT	6
NCGIDTS	7
NSC	12
NSCW	10
NTW	8
MAXCYCLE	200
CRGTBLSIZ	700
FTSIZE	700
NFTSEG	15
BLANK	55B
YES	1
NO	0
ANY	'ANY'
OFFLOAD	'OFFLOAD'
RETROGD	'RETROGD'
UNINTER	'UNINTER'
INTER	'INTER'
HOSA	'HOSA'
MANL	'MANUAL'
UNDEF	'UNDEFINED'
ERR	'ERROR'
BETWEEN	'BETWEEN'
WITHIN	'WITHIN'
LSHB	'LSHB'
SBBG	'SBBG'
EXCLUDE	'XCLD'
EOF	-1
BOG	-2
DELY	-3
SEA	-6
LAND	-11
OK	-12
DISCHARGE	-13
RECEIVE	-14
LANDV	-15
LIGHTER	-16
NSSC	-17
STOR	-18
*EOR	
*EOF	

AD-A131 715

JOINT LOGISTICS-OVER-THE-SHORE (LOTS) MAIN TEST
AUTOMATED DATA BASE REDUCTION PROGRAMS(U) ORI INC.
SILVER SPRING MD H CASEY ET AL. MAR 79 ORI-TR-1477
MDA903-75-C-0016 F/G 9/2

UNCLASSIFIED

66
NL

INFORMATION

INFORMATION

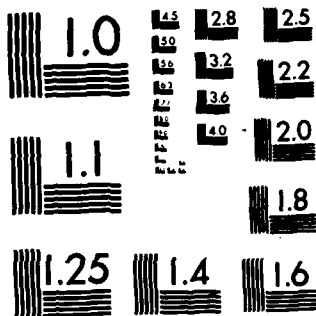
END

DATE

FILED

SEP 4

DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

SUPPLEMENTAR

INFORMATION

AD 4131715

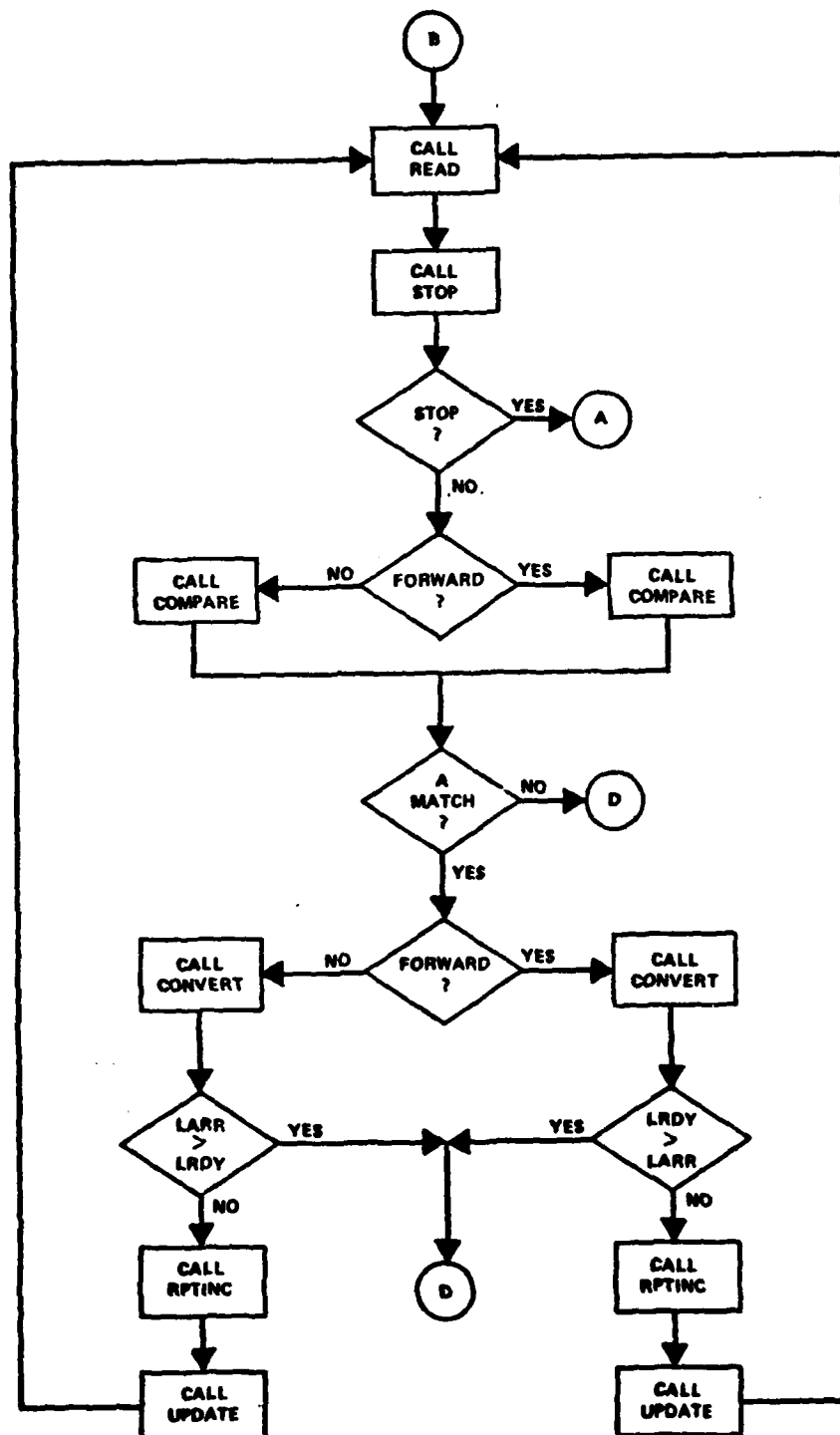


FIGURE 3.12. (CON'T) MAIN PROGRAM (SHRGEN) FLOW DIAGRAM

